October 1990

Report No. STAN-CS-90-1334

PB96-149877

Soft Configurable Wafer Scale Integration: Design, Implementation and Yield Analysis

by

Miriam Greta Blatt

Department of Computer Science

Stanford University

Stanford, California 94305

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1990	Thesis

4. TITLE AND SUBTITLE
Soft Configurable Wafer Scale Integration: Design, Implementation and Yield Analysis

5. FUNDING NUMBERS
ONR - N00014-87-K-0828

6. AUTHOR(S)
Miriam Greta Blatt

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Stanford University
Computer Science Department
Stanford, CA 94305

8. PERFORMING ORGANIZATION REPORT NUMBER
STAN-CS-90-1334 and
CSL-TR-90-445

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
DARPA
1400 Wilson Blvd.
Arlington, VA 22209

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Soft-Configurable Wafer Scale Integration uses software controlled switches to connect up the fault-free parts of a wafer. Compared to hard configuration, the soft configurable approach has the advantages of providing low-cost connections and runtime fault tolerance. The dissertation describes how to achieve soft configuration with high performance, presenting a pipelined memory system implemented using this approach. The yield of the prototype is evaluated in two phases. Fault simulation applies measured defect statistics to the layout to predict the yield of each circuit unit. These unit yields are combined to produce wafer yields using redundancy models appropriate to wafer scale integration. The redundancy models constrain wafer yield by system requirements such as the minimum number of working circuit units, and whether these working units are distributed evenly around the wafer. Choice of redundancy model significantly affects the resulting wafer yield.

14. SUBJECT TERMS			15. NUMBER OF PAGES: 123
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

SOFT CONFIGURABLE WAFER SCALE INTEGRATION: DESIGN, IMPLEMENTATION AND YIELD ANALYSIS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

By
Miriam Greta Blatt
June 1990

Abstract

Soft-Configurable Wafer Scale Integration uses software controlled switches to connect up the fault-free parts of a wafer. Compared to hard configuration, the soft configurable approach has the advantages of providing low-cost connections and runtime fault tolerance. The dissertation describes how to achieve soft configuration with high performance, presenting a pipelined memory system implemented using this approach. The yield of the prototype is evaluated in two phases. Fault simulation applies measured defect statistics to the layout to predict the yield of each circuit unit. These unit yields are combined to produce wafer yields using redundancy models appropriate to wafer scale integration. The redundancy models constrain wafer yield by system requirements such as the minimum number of working circuit units, and whether these working units are distributed evenly around the wafer. Choice of redundancy model significantly affects the resulting wafer yield.

Acknowledgements

I would like to thank Manolis Katevenis, whose ideas started this work, and Mark Horowitz, who patiently helped me along the way to implementing and testing the prototype. John Hennessy supported and encouraged me throughout, giving prompt advice and feedback directed towards putting it all together.

Technical advice and encouragement were gratefully received from Wes Lukaszek, Russell Kao, John Acken, and Neil Wilhelm.

Margaret Rowland and Darlene Hadding helped with clerical matters, keeping all the appropriate forms flowing smoothly. Charlie Orgish and Laura Schrager kept the computers running, fixing hardware and software problems promptly and cheerfully. Financial support was provided by DARPA contract N00014–87–K–0828 and a Fellowship from Xerox Corporation.

Family and friends helped keep me going, and provided a reason for finishing ("when will you be done??"). I especially wish to mention my father, John Blatt, who wrote two PhD theses in half the time it took me to complete one, but unfortunately did not live to see me graduate.

Most of all I thank my husband Malcolm Wing, who stood by me throughout to help whenever I needed it.

Contents

Chapter 1

Introduction

Wafer scale integration (WSI) achieves high circuit density by a large increase in die area — using the entire wafer as one huge die. Though frequently dismissed as radical, WSI could become the conservative approach to faster circuits, after device scaling reaches physical limits, but before any future technology is viable (such as quantum-level devices).

High circuit density leads to lower costs and higher reliability. Combining functions onto one die can reduce capacitance, thus increasing performance and reducing power dissipation. These motivations drive the integration of larger circuits onto single dies.

Researchers have pushed integration beyond single dies to entire wafers for many good reasons. Foremost among these is the potential to build a sizable system for the cost of only one wafer. WSI reduces dicing and packaging and printed circuit board costs, and can provide high bandwidth among parts of the wafer, bypassing the restrictions of external pins. Wafer scale circuits are more reliable, as one wire across a wafer fails less frequently than multiple connections among separately packaged dies on a printed circuit board.

J. Brewer presents reliability estimates in *Wafer Scale Integration* ([13], pages 17–23) predicting WSI failure rates 32 times lower than for a similar design using conventional board and packaging technology. Brewer also predicts a drop in power requirements by

a factor of 5 over conventional technology due to the greatly reduced number of chip output drivers. He compares a 10cm diameter wafer with 450,000 gates against a board with 66 VLSI gate arrays (each with 7500 gates) that includes 6 extra VLSI parts for glue logic. Conventional VLSI reliability models are extended for WSI prediction. Brewer predicts that 95% of the VLSI failures are due to packaging. He chose not to include extra gates in the wafer for fault tolerance. However, even with a pessimistic assumption of 100% redundancy (900,000 gates), the wafer is still over 10 times more reliable than the board, according to Brewer's reliability data.

WSI could derive a cost benefit from staying just behind the leading edge in device sizing. Lithography costs have risen exponentially for over 25 years[36], becoming an ever increasing portion of the total cost of each chip. State of the art fabrication equipment will cost $1 billion by the mid–'90s, with lithography accounting for 25% of the total. Improving density through large increases in die size may be cheaper.

The primary constraint on increasing die size is the rapid decrease in yield. For a circuit the size of an entire wafer, yield will be nonexistent. WSI circuits are unmanufacturable without the addition of special mechanisms to bypass fabrication defects. These additions cost both in design time and silicon area.

Companies such as Texas Instruments, Hughes Aircraft, McDonnell Douglas and IBM worked extensively on WSI in the mid 60's to late 70's [27, 37, 9, 15]. While many of these and more recent projects produced working prototypes, none could compete with the cost benefits of scaling to smaller device sizes. Over the years researchers have gradually reduced WSI costs with the development of better and cheaper fault tolerance schemes. The most notable success is the now common use of spare rows and columns in RAMs, greatly increasing yield with low area overheads. Spare rows and columns increase the size of the RAM array that can be built with high yield, but do not protect against faults in the surrounding logic (decoders and sense amplifiers) as needed for a full wafer device. Random logic requires much larger area overheards for fault tolerance, sometimes exceeding 200%.

Multichip carriers provide an alternative to both device scaling and WSI, achieving high circuit density by packaging many dies very close together on a common substrate.

Die yields are separated from substrate yield, since only functioning dies are used. Developing this technology is very expensive, and it will not be as reliable as WSI because of the many connections between die and substrate and within the substrate. Multichip carriers are probably the most viable alternative to device scaling for many applications, so we will review examples of multichip carrier technology in chapter 6, and discuss their prospects in chapter 7.

1.1 Research Goals

Soft configuration is a method of providing fault tolerance that can be used to bypass both fabrication defects and faults that occur during system operation (runtime faults). This thesis describes the soft configuration technique, and presents a prototype design built to investigate it. We evaluate the prototype, developing yield models appropriate to wafer scale integration.

The remainder of the chapter clarifies these issues: the goals of runtime fault tolerance, methods for masking circuit defects, and wafer scale yield models. We conclude by previewing the remainder of the thesis.

1.2 Runtime Fault Tolerance

The idea of designing with runtime fault tolerance predates WSI, going back to the very earliest computers, built from vacuum tubes that failed regularly. Component costs were prohibitive, so extra components for fault tolerance were not included. With the introduction of the transistor, computer parts have become cheaper and more reliable. System failure is now more costly than redundancy for many applications.

The primary goal of runtime fault tolerance is continuous operation despite faults. Methods for achieving this vary depending on the required system lifetime before failure, and the amount of downtime that can be tolerated. Existing systems fall into three categories:

long-life: Unmanned spacecraft and satellites frequently require correct operation over a period of several years with little or no possibility of external repair. However, unlike the other systems, long-life applications may be able to tolerate downtime of as much as a week, provided that the system is fully functional afterwards.

high availability: Transaction systems such as banks, airline reservations, and the stock market can afford only very short downtimes before incurring losses from dissatisfied customers. The phone system also requires high availability, both to satisfy customers and to allow problem correction at regularly scheduled maintenance visits without disrupting service in between. A typical phone system requirement is a maximum of 2 minutes downtime per year.

critical computation: Operations that cannot tolerate any downtime at all require extremely high levels of fault tolerance. One example is computations during the take-off period for the space shuttle.

An excellent general introduction to this field can be found in *Design and Analysis of Fault Tolerant Digital Systems* by Barry Johnson[23].

Systems presenting the illusion of continuously correct operation must detect and diagnose their own errors, and recover without losing data. These requirements for runtime fault tolerance add to the overheads already needed for WSI. Additional circuitry for self-diagnosis and recovery takes area from the application, and provides more opportunities for faults. Thus, some balance is required to provide runtime fault tolerance using WSI.

Runtime faults can be bypassed using any of a multitude of methods. We consider only those that can also be used for yield enhancement.

1.3 Fault Masking Methods

Fault masking uses spare circuit units to replace those containing defects. Faulty units are ignored or disconnected. We use the term **configuration** to refer to the process of

establishing a network of fault-free connections among the fault-free circuit units. Such circuit units could, in theory, be as small as a single gate, but in practice the gain in being able to configure every working gate is offset by the large overhead and lower yield of all the configuration circuitry. Hence, we assume that each circuit unit is of size comparable to a die, and refer to it as a **site**.

Circuit faults can be masked using one or more of the approaches presented below. In a **hard-wired** design, wires are physically altered to connect the good parts and disconnect the others. With **soft configuration**, connections are established by switching transistors set by external software. A **voting** scheme uses replicated circuit units, and voting circuits to forward the majority results.

A **hard-wired** scheme binds the configuration at the earliest possible point — namely the last step of fabricating the wafer. One example is the Restructurable VLSI project at MIT Lincoln Labs[39]. They use lasers to both make and break connections. Hard-wired connections have the advantage of relatively low resistance, but require extra time and equipment for restructuring. Hard-wired restructuring cannot be performed while the system is running, so it cannot provide runtime fault tolerance.

With **soft configuration**, connections are switched by transistors under external control. The configuration is bound at power-up, but can be changed at any time. Fault detection, diagnosis and reconfiguration interrupt normal operation, but only for a few milliseconds. Soft masking will be described in detail, as it is the basis for the system described in this thesis. Transistor connections are highly resistive, restricting applications to those that can tolerate the transistor delays, while still providing good performance. Such applications profit from yield improvement and runtime fault tolerance, without requiring either large silicon area overheads or special purpose laser equipment.

Firm configuration is a variation on soft configuration with the switch settings stored in non-volatile memory. The configuration is bound at power-up, so runtime faults cannot be corrected without turning off the system. An example is presented in [22].

Voting designs bind the configuration at the latest possible moment, when each function is performed during normal operation. Such late binding is needed at some level

(possibly software) for any system designed to handle critical computations. The Trilogy project[33] used triple modular redundancy (TMR), where functional units are replicated 3 times. The area overhead for TMR is considerably more than 200%, as in addition to the area needed for 3 copies of each functional unit, extra space is needed for each voting unit, and for wiring. If common-mode failures are to be avoided, the 3 copies must be spaced across the wafer, requiring even longer wiring than is already needed to cover the extra space taken by tripling the area. This is turn leads to larger drivers and wider wires to lower the resistive impact of long lines, hence to higher costs and lower performance. Trilogy was not able to make a commercially competitive system using TMR, however several (noncommercial) space-borne projects have used it. The key advantage is the ability to function continuously despite faults, whether intermittent or simply not present immediately after fabrication.

Many systems use some combination of the above. Although presented solely as hardware methods (required for WSI), soft configuration and voting can also be implemented using software if the only goal is runtime fault tolerance[23].

All these methods negatively affect system performance in various ways (more capacitance, higher resistance, longer lines). System performance increases only if the integration onto one piece of silicon (avoiding package and board crossings) outweighs the negative effects of supporting fault tolerance and reconfiguration. For each design, the required yield, performance and reliability can be traded against each other to reach an acceptable balance.

In summary,

- Hard-wired schemes increase yield with low resistance connections, but require extra equipment and fabrication steps and cannot be used in the field for runtime fault tolerance.

- Voting provides both yield improvement and runtime fault tolerance but with large area overheads, and much longer interconnect wires.

- Soft configuration provides runtime fault-tolerance, without the excessive costs of

voting, and yield improvement, without expensive special purpose physical alterations. However, soft (and firm) configuration rely on connections through resistive transistors that add delay.

Every method for masking faults has a weak point with global signals, as a single fault can potentially jeopardize the correct functioning of the entire wafer. Yield loss due to global signal failures can be very costly.

Another common concern is faults in the configuration circuitry. With laser configuration, two paths can be supplied for each connection, so a single error can be corrected. With TMR, the usual solution is triplication of the voters as well as the functional units, thus passing on voter error as an error in the following functional unit. Soft configurable switches must be able to disregard the outputs of neighboring switches as well as sites.

1.4 What is Yield?

Having described some ways of using redundancy to improve yield, the next step is to evaluate that improvement, so that alternatives can be compared at design time.

The yield of a production line is the fraction of parts produced that can be used. When the production line produces diced wafers (also known as silicon chips), yield is the fraction of dies that function correctly. When, instead, the production line produces fault-tolerant WSI wafers, yield is the fraction of fault-tolerant WSI wafers that function correctly. The difference between die and wafer yields arise from the definition of correctness.

What does it mean for a fault-tolerant WSI wafer to function correctly? It cannot be that every site, every switch, and every wire be 100% perfect, as this would require no use of the fault-tolerant circuits, and provide 0% yield (any production line that good should be moving to a smaller linesize, so long as scaling remains feasible). Since a fault-tolerant WSI wafer is designed with the assumption that it contains faults, its yield must be defined as the fraction of wafers produced that contain a sufficient number of working sites to configure a working system.

These terms "sufficient" and "working system" are defined by the application, so are not fixed at fabrication time, nor can they be defined in a general way. Thus, a production line cannot produce fault-tolerant wafers with a definite fixed yield. While one user can construct a working system from, say, 50% working sites, another may require more. Furthermore, depending on the configuration scheme, it may be important which 50% of the sites are working, particularly if there is more than one cell type. These sources of fuzziness lead to widely varying fault-tolerant wafer yields for the same site yields, as demonstrated in chapter 5.

The expected number of working sites relates to the average site yield, while the (smaller) number required to form a "working system" reflects the expected variance in site yield. High "working system" numbers reduce wafer yield by eliminating wafers with some, but not enough, working sites. Low "working system" numbers waste extra working sites if the application cannot use them. The expected number and variance of working sites are thus vital inputs to the process of obtaining fault-tolerant wafer yield.

Unfortunately, reliable site yield numbers (equivalent to regular yield of diced wafers) are not easy to acquire. With technology changing every year, there are no fixed defect statistics, only learning curves rising from zero yield to some reasonable quantity (if the company is to survive) and eventually very close to 1, by which time new technology is on the way. Yield prediction is reliable only for stable technology, which is always out-of-date. Only reaching fundamental physical limits will change this.

1.5 Synopsis

The primary contributions of this research are the implementation of a soft-configurable memory system, demonstrating the viability of this approach to soft configuration, and its evaluation using fault-tolerant yield models, showing how yield is predicted, and what range of yields are expected for the prototype.

Chapter 2 introduces soft-configurable WSI, and chapter 3 follows with the design and implementation of a soft-configurable WSI memory system. Chapters 2 and 3 present

work done with Manolis Katevenis[26]. Professor Katevenis worked on the initial approach to soft configuration, while I did almost all the layout, and followed it by simulating and testing the fabricated parts.

In chapter 4, fault simulation with fabrication defect statistics predicts the site and switch yields of the prototype directly from the layout. Chapter 5 derives wafer yields using various system models. Wafer yields vary enormously for different model requirements.

Chapter 6 reviews several recent WSI projects, and in chapter 7 I summarize my results and speculate on the future of WSI.

Chapter 2

Soft Configurable WSI Design

A soft configurable WSI system contains switches and sites. Sites hold the application circuits. Switches control the routing of data through the switch network to fault-free sites. The switch circuit chooses an input by setting control lines for the interfaces between the switches, deciding which fault-free neighbour switch or site will drive the switch bus.

Switch circuitry is not immune to faults, so switches must be protected from one another. Sites also need protection from faulty switches: it must be possible to access each site through multiple switches so there is choice in the event of switch failure.

Fault tolerance to improve switch yield suffers from area and delay overheads and longer design time. Switches already add delay, area and design time to the application. If, instead, switch design is simplified as far as possible, while retaining configuration capabilities, switch yields can exceed 99% as described in chapter 4.

These assumptions guide the design of soft configurable switch networks. We look at the overall layout of switches and sites, and examine the internal components of each switch. We also consider interfaces, both between switches, and between a site and the switches surrounding it.

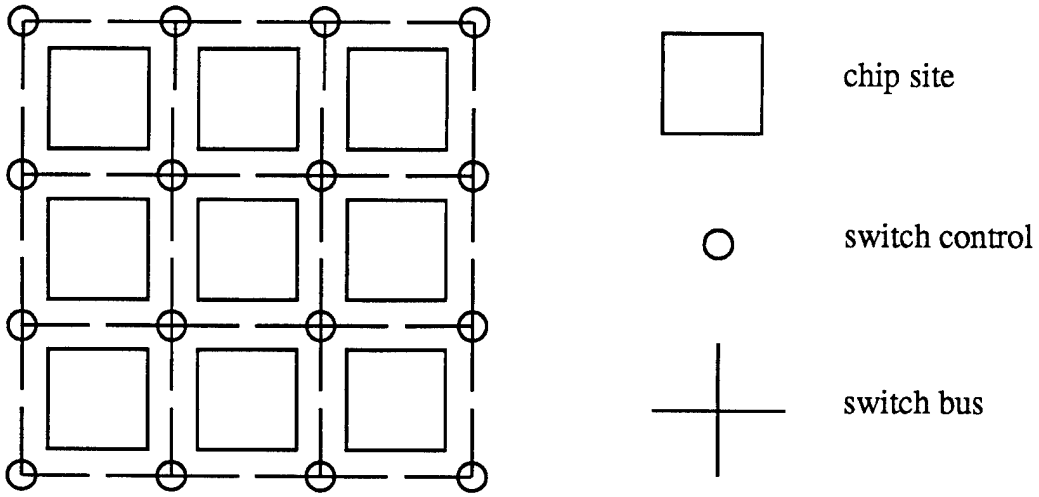


Figure 2.1: Switch positioning. Rectangles are chip sites, crosses are switch buses, and the circles contain switch control circuitry. Space between adjacent switch buses contains the switch to switch interface circuit. Space between a chip site and adjacent switches contains the switch to site interface circuit.

2.1 Connecting up Switches and Sites

Spare switches and sites replace defective units. Multiple spares per unit increase the probability of a successful configuration. Multiple units per spare reduce overheads due to spare switches and sites. Switch and site connections determine the number of units per spare, and spares per unit. More connections add flexibility, but also add fault-prone wires and controls that take area from the application.

By connecting sites only to switches, and not to other sites, we allow the switch network to use any site as a backup for any other. Switches must rely on neighbors for backup, since each site requires at least some working switches nearby for access.

Spares need not be distinguished from primary units. A uniform design where each site and switch can be used as either a spare or a primary is flexible and simplifies layout. Regular layout can use step-and-repeat lithography, with the most recent technology. The unit of replication must contain at least one site and one switch. Using no more than one switch per site maximizes the area available for the application, and provides four switches near each site, as shown in Figure 2.1.

Global signals must be fault intolerant, since one short or open in a global can impede the functionality of the entire system. Such signals are eliminated by sending global requests stepwise, requiring only a single step per clock cycle.

Only clock and power lines must be provided globally. Soft configuration cannot be used for power, since transistors are too resistive, or for clocks, without recursively requiring more clocks to switch the system ones. Clock and power configuration cannot be part of the overall approach to soft configuration, so further discussion of their implementation is deferred to section 3.6.

Repeaters used to forward signals can be readily augmented into latched stages. The latches can be used for pipelining, so that multiple items progress through the switch network simultaneously. If each site request takes several cycles, then pipelining increases the system bandwidth whenever sites operate in parallel.

Distributing global signals in stages has several effects. It provides fault-tolerant globals, as each signal is sent through independent steps that can be rerouted to bypass faulty circuitry. Faults are isolated, eliminating catastrophic defects. Each stage can boost the signals, which is important for CMOS, where delay depends on the square of the length. Repeaters can be adapted to include latches for those systems that profit by pipelining. However, latency increases relative to a single global line, as the optimal interval for signal boosting is likely to exceed the distance between neighboring switches.

Figure 2.1 shows switches at the corners of sites. They could instead be placed along the site sides, exchanging locations with the switch interface. Those interfaces that send data in and out of nearby sites then require site I/O buses routed through the corners of each site. This would bring considerable wire congestion to these corners, and is not likely to lead to a very compact layout. Placing the interface along sides of the site allows the I/O lines to be spaced so as to exactly match the site interface requirements. Placing switches at site corners leads to the use of cross shaped bus nodes as shown in Figure 2.1.

Switch interface complexity can be reduced by connecting fewer switches to each site. If 2 of the 4 nearby switches are connected then sites are still reachable in the

presence of single switch faults. This increases switch and interface yield at the expense of runtime flexibility. Losing one switch to yield failure leaves the site neighbors without switch backup at runtime. However, runtime fault tolerance requires the maintenance of site backups containing the same data; these can be used both for site failure and for failure of a switch connecting to the original site.

2.2 What does a Soft-Configurable Switch consist of?

The primary purpose of the switch is to route data between fault-free neighbors (switches and sites). So each switch must store information identifying faulty neighbors. If gates are used between switches (for electrical isolation, and to boost signals) then this information is needed only for those neighbors supplying inputs. Neighbor status can be stored as one mask bit per neighbor, used as inputs to the routing decision circuit so that only fault-free switch and site inputs are enabled. Configuration is then the process of setting the appropriate mask bits (possibly different) for every switch.

Figure 2.2 presents a switch block diagram. Full two-way communication is shown on every possible link to nearby switches and sites. The implementation of the next chapter reduces this from 8 inputs (4 switches and 4 sites) to 5 inputs (3 switches and 2 sites). This reduces the number of mask bits per switch, and simplifies control for the switch to switch interface.

Routing paths can be established either for per item (packet switched) or for an entire communication (circuit switched). The choice depends on the application: if sites are internally pipelined, then creating circuit links dedicated to feeding them may be worthwhile. The one time initialization cost must be traded against the cost of including routing information in every packet. Circuit switched links are dedicated to one communication to the exclusion of others; packet switching allows multiple transfers to be interleaved. If site operations take several cycles, then packet switching is preferable, so those cycles may be used for transfers between other switches and sites. Packet switching is assumed here, however an application better suited to circuit switching is described briefly in chapter 7.

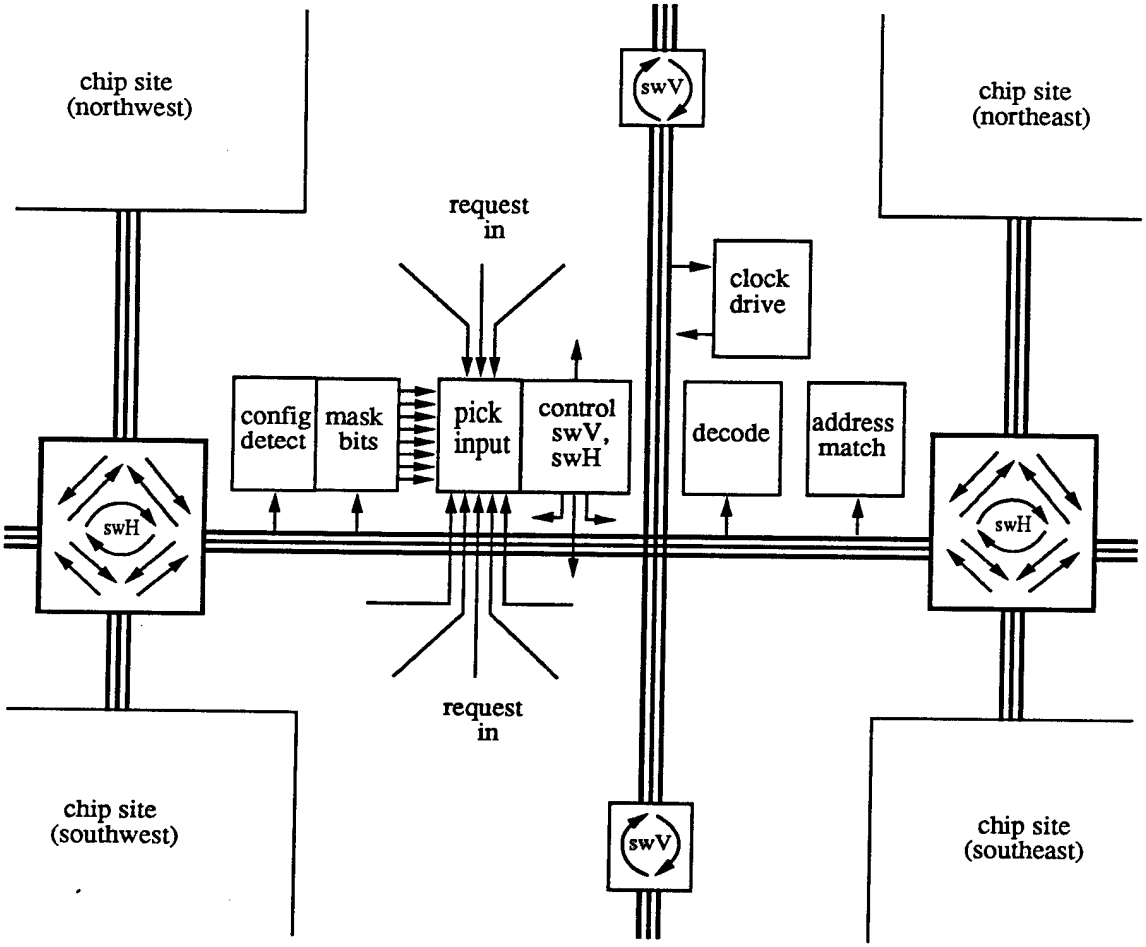


Figure 2.2: Switch block diagram. Note the cross shaped switch bus, represented by the 3 parallel bold lines in the center. The eight request lines from the four neighboring switches and four neighboring chip sites are ANDed with the eight mask bits to disable faulty inputs. Some routing choice directs the control of the vertical and horizontal switch interfaces (swV and swH) to choose which of the fault-free inputs will be read on the next cycle. The chip site interface is arbitrarily chosen to be at the top and bottom, through which paths exist to all four neighbor switches.

While good software will ensure that multiple valid inputs never reach a switch simultaneously, a safe design requires some plan for collisions. Either one input is stored an extra cycle, causing congestion in the switch network, or the unselected input is dropped (to be sent again by the external software). Whichever input is chosen, some time is needed to set the appropriate input controls on the switch interfaces; for a packet switched design, input controls must be set every cycle. This can be implemented by sending request signals one phase ahead of the data.

Routing requests to the correct destination requires either an address or a list of directions. Such directions could be included in the command, and stripped off by each node it passes through[32, 4]. However, a faulty switch might destroy the directions or send them the wrong way. The alternative shown in Figure 2.2 assumes that each switch and site has its own address so will recognize commands addressed to it.

Correct routing depends on the integrity of the mask bits. So it is very important that the most likely faults (shorts where the bus lines cross each other) can be detected by the circuit that configures the mask bits (shown in Figure 2.2 as "config detect"). If all the mask bits are low on power-up, then switches will ignore every input (assuming the neighbors are faulty). Bypassing this barricade to set the switch masks irrespective of current settings is a nontrivial exercise.

One could bias the mask bits to power-up high. However, a faulty switch could corrupt a configuration command so that a good switch is treated as faulty, and it would be very difficult to detect or correct this. Without global controls, mask bits cannot be set directly from external signals. Instead, a redundant coded sequence can be sent to each switch, starting with those at the wafer edge, and proceeding sequentially inwards. Switch buses can be used for mask configuration as well as for site data.

Efficient parts testing requires some means of ensuring that the data is flowing through the specified switch path without having to probe internally.

To summarize, each switch requires these components:

- **addressing logic:** identifies commands for this switch

- **command decode:** distinguishes between switch commands, and separates them from site commands
- **mask bits:** masks faulty input sources (switches or sites)
- **routing control:** chooses which fault-free input to receive
- **mask configuration:** sets the mask bits, guarding against configuration requests from faulty neighbors
- **clocks and power:** safeguards against faults for these important global signals
- **testing:** verifies the configured switch path both at configuration time, and during regular use to check for runtime faults

2.3 Switch to Switch Interface

Data routing can be provided by pass transistors (or transmission gates). However, they would neither boost the signal nor provide safety from neighboring power shorts. Both of these concerns can be addressed by using tristate inverters to amplify the signal and electrically isolate the switch buses. Tristate inverters are needed for each possible link for each signal within the bus. These links are shown in Figure 2.2 as arrows inside swV and swH.

Layout could be a problem if the interface cells are stacked in a row, blowing out the height (or width for swV) of the bus. Instead, cells are placed diagonally, requiring only one cell height extra beyond the switch bus at any point. Cells that link with the site (those of swH in Figure 2.2) can be spaced out along the site side so as best to match the location of site inputs and outputs. This diagonal organization is shown in Figure 2.3.

Power separation is vital to ensure that shorts in one switch or site do not disable functioning neighbors. Tristate inverters controlling bus inputs to the switch must be powered from that switch. It is not sufficient to electrically isolate the bus nodes if the power lines remain connected. A bidirectional switch interface needs two sets of power lines, one from each receiving switch.

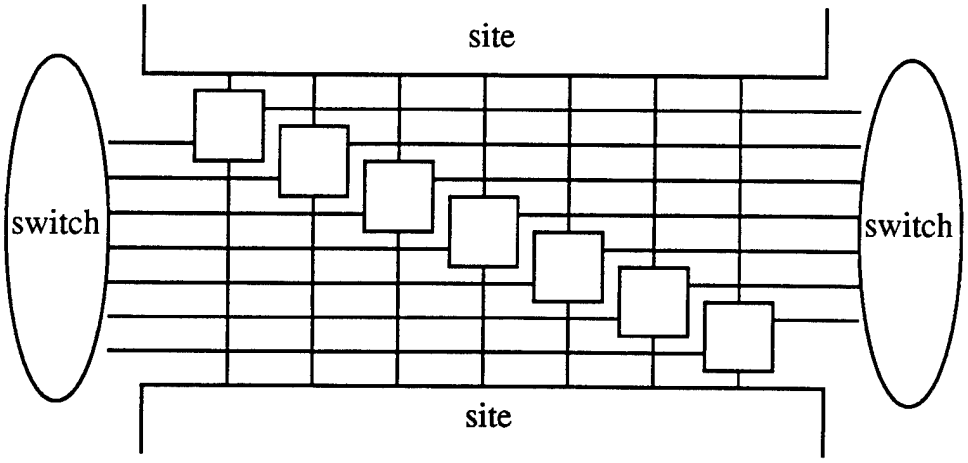


Figure 2.3: Diagonal organization of the switch to switch interface.

To summarize, the switch interface needs

- **electrical isolation:** electrically separate bus nodes
- **power net isolation:** separate power lines for each switch receiving input
- **drivers:** reinforced signals for good performance
- **diagonal layout along site side:** compact physical layout that matches the locations of site inputs and outputs

2.4 Switch to Site Interface

Sites must be able to ignore faulty switches. This can be implemented in a similar fashion to the switch masks, using one mask bit for each switch that provides inputs to the site. Like the switch, site masks need special initialization, since on power-up they may be set to ignore every input. Sites do not need mask bits for other sites, as data flowing from one site to its neighbor always passes through a switch.

Each site needs an address; this can be soft settable, since site failures do not affect the switch network connections. Sites must generate request signals ahead of sending

out data to the switches, just like the switch request signals. Sites also decode bus data to check for commands addressed to them.

Like the switch to switch interface, the switch to site interface must have electrical isolation of bus nodes. This prevents leaks of bus data from one switch to another through the site interface. Power net isolation is needed, so that shorts in the site do not affect neighboring switches. Power net isolation can also be used to power down sites with faults, or sites that are not needed for the application, reducing the system power requirement. Isolation is provided by feeding in separate power lines, and using tristate drivers for every signal coming into and going out from the site.

So a switch to site interface needs

- **mask bits:** ignores faulty switches
- **mask initialization:** sets mask bits
- **site address:** distinguishes commands for this site
- **decode and compare unit:** monitors switch bus for commands
- **electrical isolation:** tristate drivers for inputs and outputs
- **power net isolation:** separate power for each site

2.5 Review of Soft Configurable WSI Requirements

There are some key issues in designing a soft configurable switch network. They include fault independence, manufacturing with step and repeat, and initializing the network.

We use the phrase fault independence to mean isolation of faulty parts, where circuit units are sufficiently independent so that no single fault can disable more than that one unit. Fault independence is crucial for damage containment, restricting the effects of each fault to as small an area as possible. Sites must be able to interact with multiple switches, so they are not dependent on particular ones to communicate their inputs and outputs.

Switches must be able to completely ignore inputs from faulty switch and site neighbors, functioning correctly through alternative routes. Global signals must be eliminated as far as possible, by sending controls through the network one step at a time, using the switch network to route them safely rather than relying on a single widespread net.

Step and repeat manufacturing lowers fabrication costs by providing higher yield. It requires repeatable subunits, thus a regular array with few site types and minimal specialization for array edges.

Initializing the network is a key step for providing fault independence. It requires some means of testing the switches, and detecting which ones have faults. Fault-free switches must be able to ignore initialization requests from faulty neighbors.

Another important task of the switch network is routing. There must be some means of addressing particular components of the network, and specifying where data flows.

Finally, all of this must be provided compactly, to take as little as possible of the area that can be profitably used by sites. It should also be designed for very high yield, so that the mechanism for evading faults is not sabotaged by its own problems.

All of these issues are examined in detail in subsequent chapters. Chapter 3 describes the implementation choices for the prototype pipelined memory system with a soft configurable switch network. Chapters 4 and 5 examine the yield of this design.

Chapter 3

Soft Configurable WSI Implementation of a Pipelined Memory System

We built a pipelined memory system to demonstrate the ideas of the previous chapter. The die photo of Figure 3.1 shows a 2x2 array of RAM sites surrounded by a 3x3 array of switches. The use of RAM for the sites was intended to leave the majority of design and implementation time to be spent on the switch network.

The prototype implements vertical switch paths that transfer data to and from the RAM sites. Addresses and data are sent from the bottom of the switch path, and travel through the switch network, bypassing faulty switches using horizontal links. Sites send read-data through a switch path to output ports at the top. Read and write latency includes the delay through the switch path. Pipelining allows several sites to operate in parallel, conveniently streaming reads and writes. This, combined with multiple switch paths that can operate in parallel, provides high bandwidth suited to a main memory refilling cache lines (with one site supplying each word in the line) or to a fast disk cache.

We examine details of the implementation below, starting with the overall dataflow between switches and sites. Mask initialization determines which neighbor switches are trusted; we show how to protect switch masks from neighbors containing shorted or stuck bus wires, while receiving valid mask requests. Masks eliminate faulty inputs from

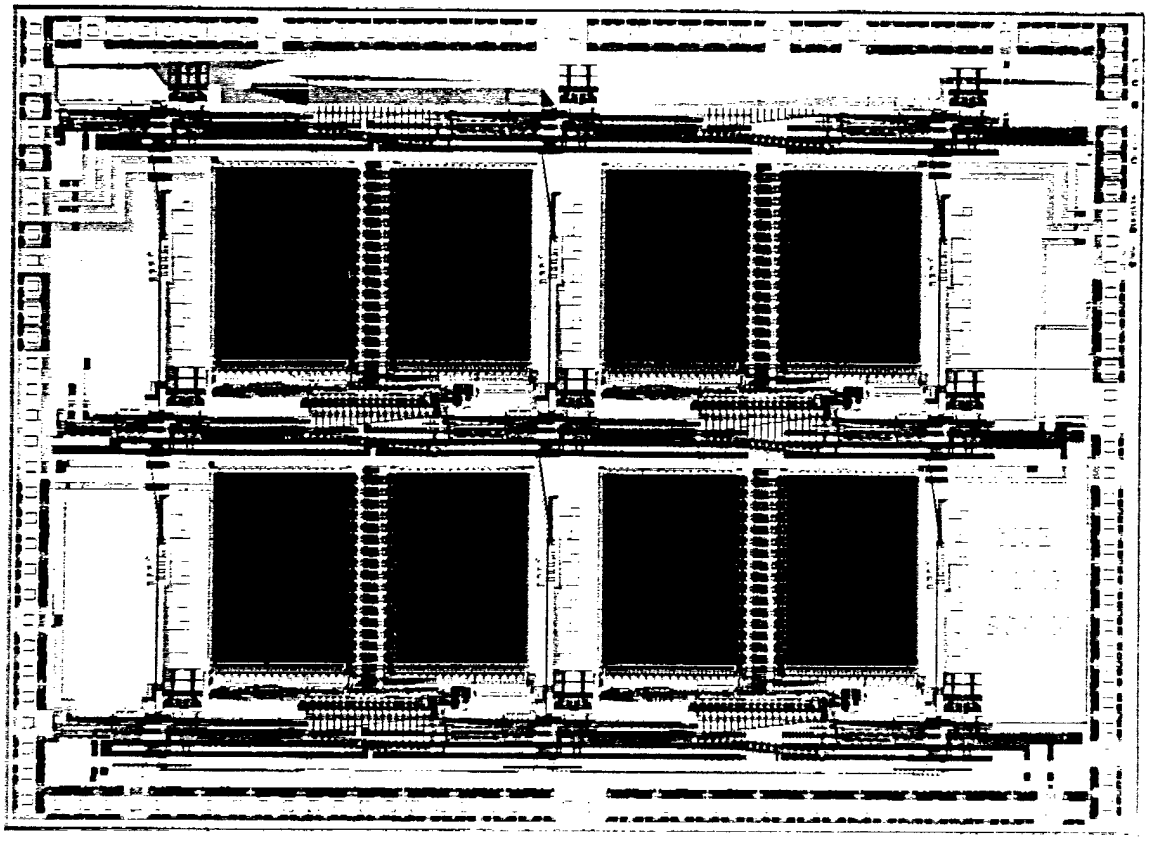


Figure 3.1: Die photo of the soft-configurable pipelined RAM

consideration, however we still need circuitry to decide which of the fault-free inputs to choose. These are the primary logic components of the switch.

Building a network requires interfaces between switches, and between sites and the switches around them. We examine the contents of these interfaces, and of the RAM sites. Clock and power signals pose special problems which are described, together with the solution chosen for this implementation. The chapter concludes with an example showing switch configuration and site reads and writes, followed by comments on the implementation and testing experience.

3.1 Global Switch Connections

A maximally flexible design with one switch per site provides links from each switch to and from all 4 neighbor switches and 4 neighbor sites, for a total of 16 I/O directions per switch. Switches have 8 mask bits for the 8 possible switch or site inputs, and sites have 4 mask bits for the 4 possible switch inputs. Fault tolerance for both switches and sites can be provided with fewer connections, trading flexibility for a simpler switch design with higher yield.

Figure 3.2 shows the global dataflow of the memory switch network. Switch and site commands enter from the bottom and flow up to the top, possibly with horizontal jogs in either direction. Sites read from a switch below (bottom left or right) and send results out through a switch above (corresponding top left or right). The high yield projections for this simple switch make extra switch connections unnecessary, and probably counterproductive (leading to lower switch yield, hence more need for connection flexibility).

With connections as shown in Figure 3.2, each switch needs 5 mask bits: 3 for switch inputs and 2 for site inputs. The sites need only 2 mask bits for the 2 switch inputs. With the further assumption that each site accepts input from only one switch chosen at configuration time, this can be simplified to a single mask bit for sites. This assumption is necessary for the checkerboard clocking described in section 3.6.2. The single mask bit chooses which of the two bottom switch neighbors to monitor for input. The mask bit of the corresponding top neighbor switch must be set to receive site outputs as the other output switch could receive only gibberish, once again for timing reasons. These restricted connections provide enough flexibility to configure around both manufacturing and runtime faults, using a small and simple switch implementation.

Figure 3.3 shows the switch in more detail, with the connections as in Figure 3.2, and correspondingly fewer mask bits. *Config* lines are used to initialize the mask bits. *Stuck detect* boxes guard against inputs that are stuck high, giving preference to those that switch. The *match* box compares the address lines on the bus with the switch address, generating a recognition signal used for switch path testing. The following

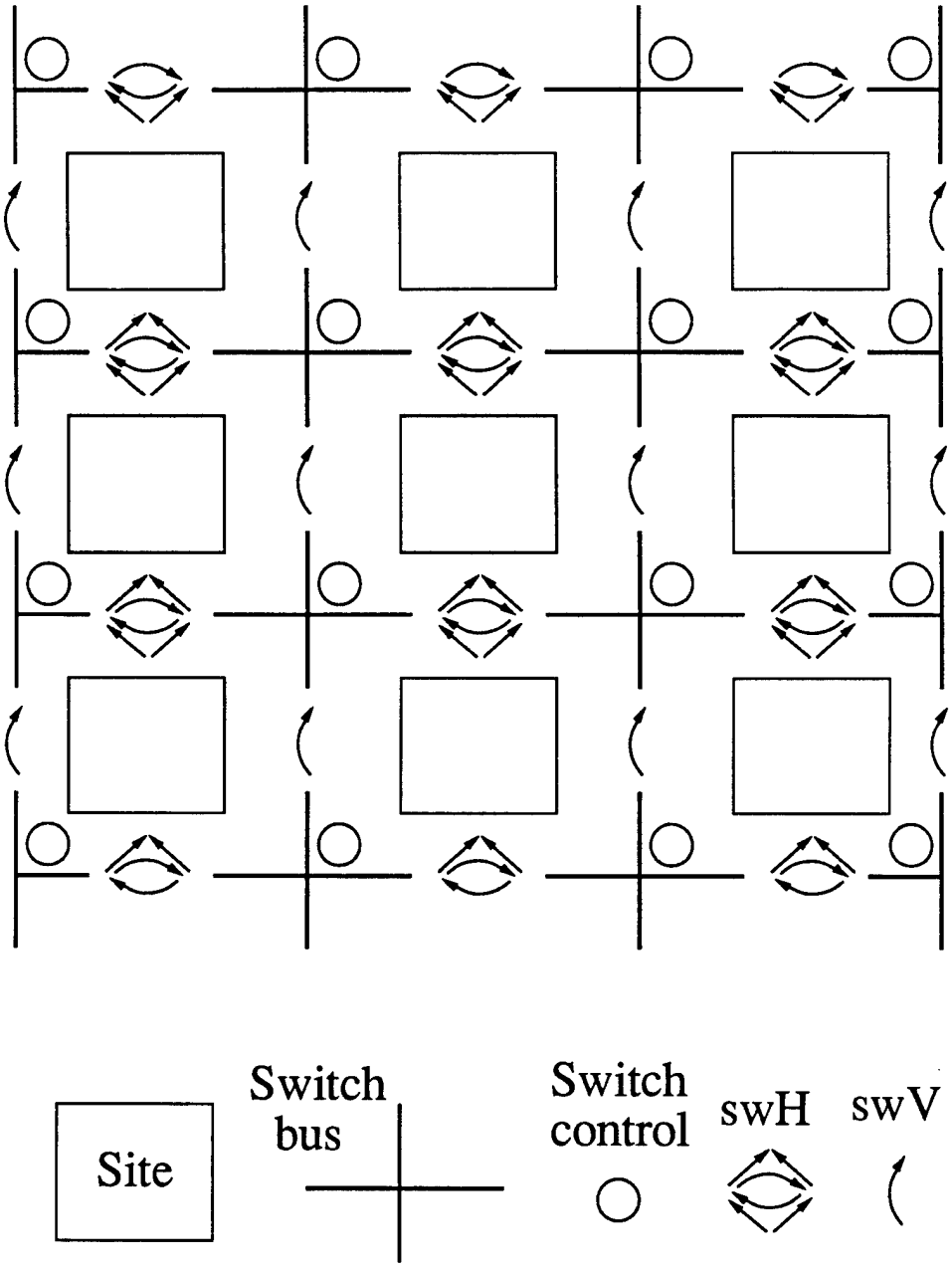


Figure 3.2: Memory switch dataflow

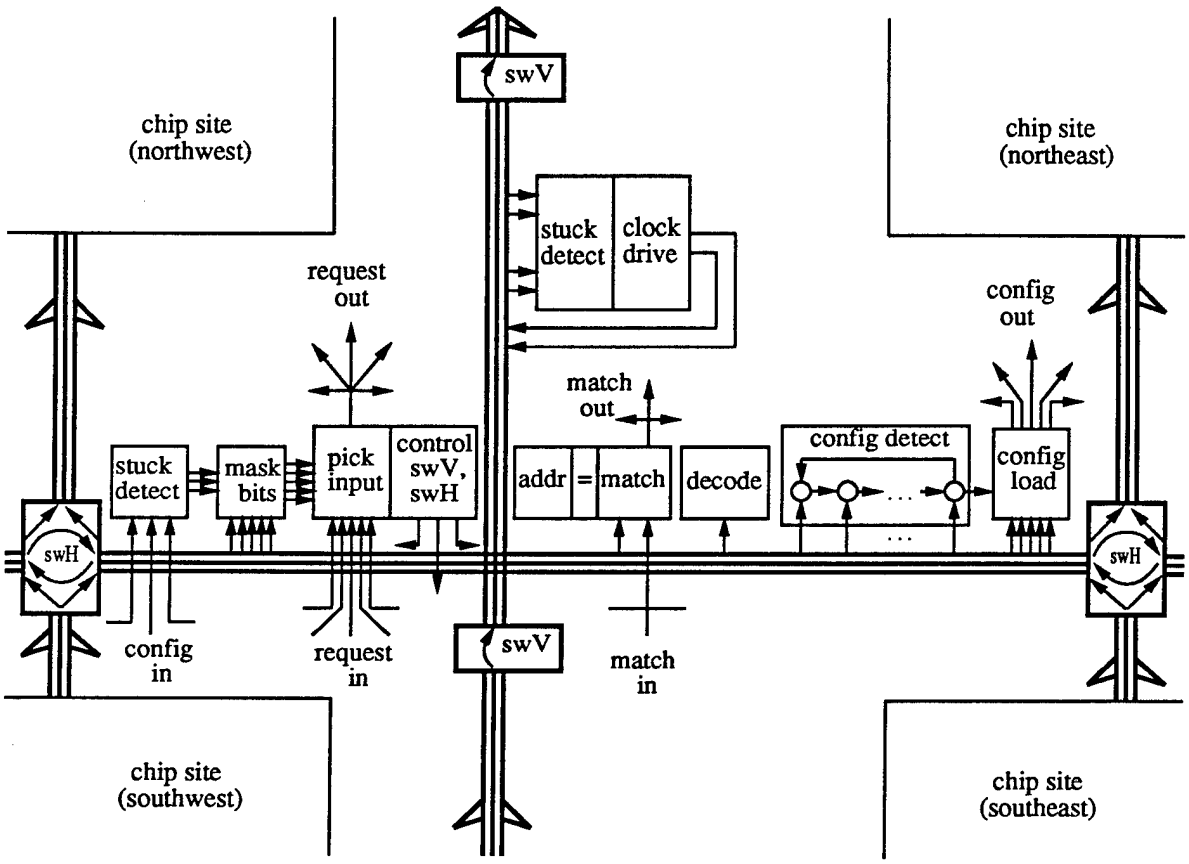


Figure 3.3: Switch block diagram. Configuration, request and address match input and output signals are directed to neighboring switch nodes and chip sites. Each arrow inside the horizontal and vertical switch interfaces (swH and swV) shows a possible direction of data flow for the entire bus.

sections examine the implementation of each part of the switch, and the switch and site interfaces.

3.2 Mask Configuration

Configuring the mask bits without using global signals can be somewhat tricky. The difficulty is that the switch node we are trying to access may have every mask bit disabled, so will ignore any configuration request sent to it. Biased circuits could initialize fully functional masks as desired. This would have to be in conjunction with some kind of round robin choice of inputs, so that switches would not focus only on a faulty neighbor. A simpler approach with hardwired input priorities requires some other means of forcing a switch to receive data from other inputs, especially when the biased default input is faulty.

Each switch and site must be able to detect valid configuration requests. One solution is to have multiple circuits watching for configuration requests from every possible direction. The equivalent function can be provided using only one configuration detector per switch as follows. Instead of attempting to modify its own mask, the switch detector forwards valid configuration requests directly to all neighbor switches and sites, bypassing their input masks.

The minimal information needed for a configuration is the switch address, and the mask bit settings for its neighbors. This information can be sent reliably using a myriad of different encoding schemes. A coding that includes the information in plain text simplifies the implementation. Since configuration will be performed only on power-up, and on the rare occasions when some circuit fails during operation, it is not a time critical operation. So rather than widening the switch command to include the redundant part, it can instead be sent in several parcels, time-multiplexed. The coding used in this implementation consists simply of rotating the plaintext configuration request (switch address plus neighbor mask bits) one bit at a time, and comparing it with a stored version rotated by the switch. The CMOS implementation of the configuration detector is shown in Figure 3.4.

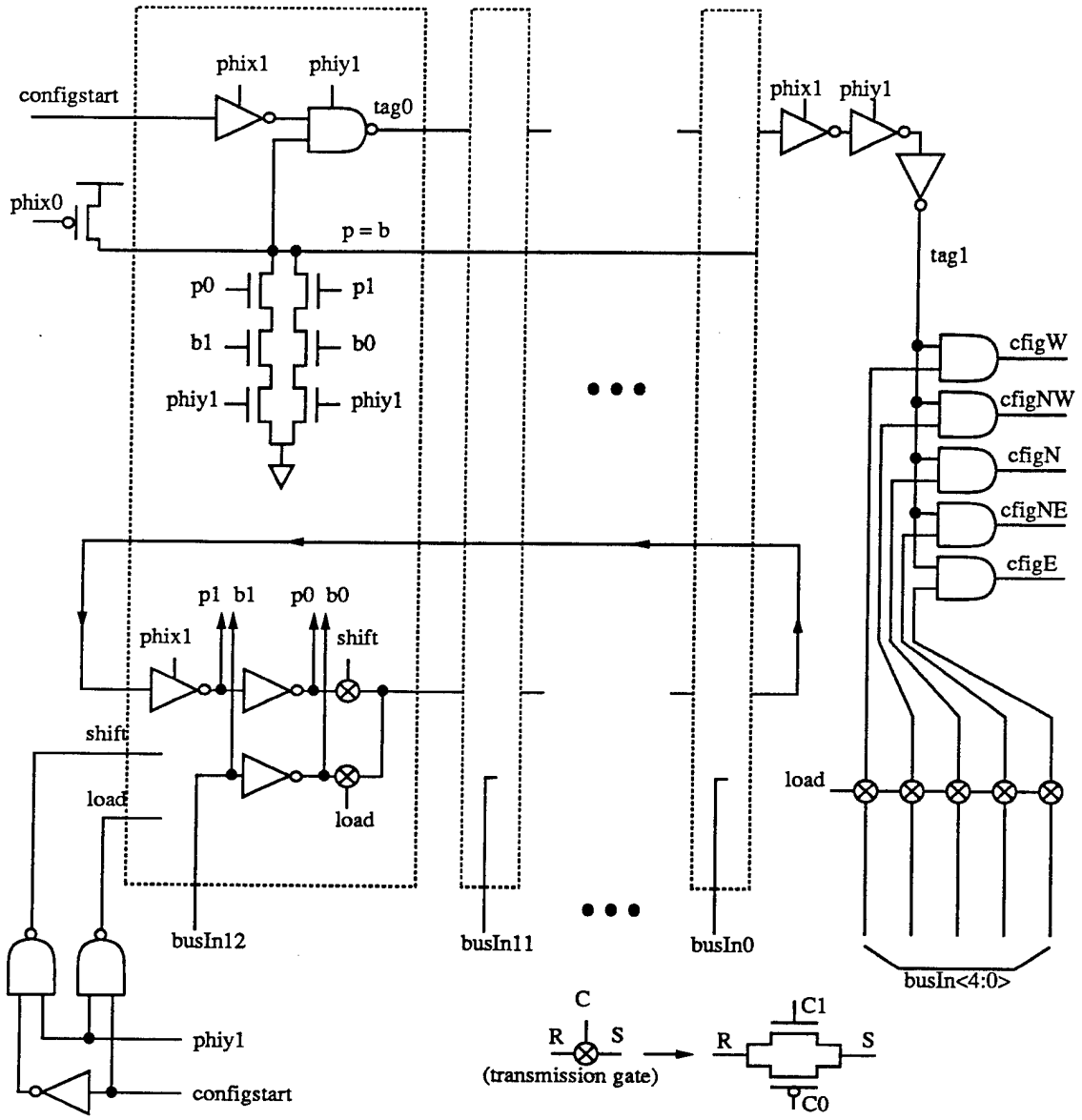


Figure 3.4: Configuration detector. On the first cycle, the bus signals are loaded in to the rotating ring (wires p0 and p1). On each subsequent cycle the p bits rotate one step, and are compared with the incoming bus signals (b bits). If every comparison works, then the tag1 signal enables the sending of the cfig signals to configure neighboring switches and sites to accept future inputs only from this switch. Signal names (other than bus lines) that end in 1 are active high; those that end in 0 are active low.

We show that this coding satisfies the primary criteria — detection of bus stuck-at and bridging faults, providing the input is of prime length and contains both 0's and 1's[26].

Let n be the length of the input word $W = w_{n-1}w_{n-2}\cdots w_1 w_0$. Then the coded transmission contains n copies of this word, each rotated by 1 bit:

$$\begin{array}{cccc} w_{n-1} & w_{n-2} & \cdots & w_1 \quad w_0 \\ w_0 & w_{n-1} & \cdots & w_2 \quad w_1 \\ \vdots & \vdots & & \vdots \quad \vdots \\ w_{n-2} & w_{n-3} & \cdots & w_0 \quad w_{n-1} \end{array}$$

Notice that every column is a rotated version of the input word W.

Theorem:

Let C and D refer to the c'th and d'th columns respectively of this coded transmission, with $c \neq d$, $c, d \in 0, \cdots, n-1$. Let $C = D$ mean that each bit of C is equal to the corresponding bit of D.

If $n = |W|$ is prime, then $C = D$ if and only if W is all 0's or all 1's.

Thus, prime n and $W \neq 0, W \neq 1$ suffices to guarantee distinct columns.

Proof:

Assume that $C = D$, and let $x = |c - d|$, so $x \in 1, \cdots, n-1$. Then

$$w_k = w_{(k+x) \bmod n} \qquad k \in 0, \cdots, n-1$$

This is because C and D are both rotated versions of W, starting x positions apart.

Setting $k = 0, x, 2x, \cdots, (n-1)x$ leads to

$$w_0 = w_{x \bmod n} = w_{2x \bmod n} = \quad \cdots \quad = w_{(n-1)x \bmod n}$$

Since n is prime and x is nonzero, these subscripts contain every integer between 0 and $n-1$. Thus,

$$w_0 = w_1 = \cdots = w_{n-1}$$

which completes the proof.

Corollary:

If wires c and d are shorted together, or any wire is stuck, then the transmission will be rejected by the configuration detector.

Proof:

Let C' and D' be the columns received through wires c and d when sending C and D respectively.

Then complementary inputs are shorted to intermediate voltages, resulting in either:

1) $C' = D' \vee C$ or $D' = C' \vee D$
2) $C' = D' \wedge C$ or $D' = C' \wedge D$
3) $C' = D'$

Cases 1) and 2) arise when intermediate voltages are interpreted differently by the receivers due to different threshold voltages. Case 1) adds 1's and case 2) removes them, resulting in C' having a different number of 1's from D'. Either way, C' is no longer a rotated version of D', so the transmission will be rejected.

If 1's are neither added nor removed, then we are looking at case 3). By the Theorem, case 3) will only be accepted if $C' = D' = 0$ or $C' = D' = 1$.

Thus, a fault-free detector will reject transmissions through shorted bus wires, provided they contain both 0's and 1's.

Stuck wires result in columns received as all 0's or all 1's, never the rotation of a valid message.

Configuration signals forwarded by a detector bypass the masking protection of the neighbor switches and sites. Thus some other way is needed to protect the mask bits of these neighbors from a faulty switch that is constantly demanding attention. The configuration line of the faulty switch is effectively stuck high. Stuck protection can be

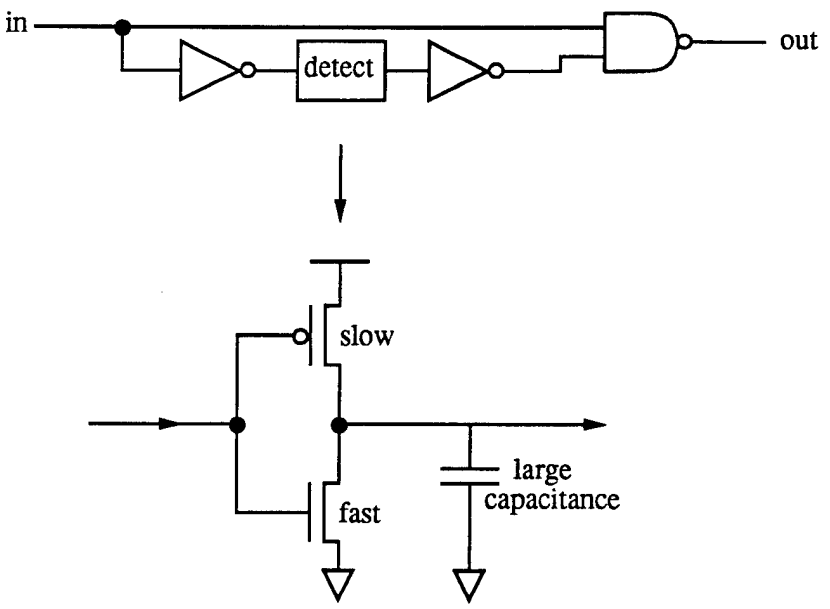


Figure 3.5: Stuck detection. The slow pullup combines with a large output capacitance to provide the required asymmetry. This inverter takes a very long time to pull high, and only affects the output of the NAND gate when high.

provided by an unbalanced inverter with a very large capacitor slowing the transition in one direction. The *stuck detector* circuit is shown in Figure 3.5. It works by pulling down configuration lines that remain high for a very long time (as determined by the capacitor), thus allowing the switch or site the freedom to receive configuration requests from other neighbors that are fault-free.

Stuck detectors work only for their inputs. A stuck line on the output of the detector can still disable a switch. Also, they do not protect against stuck-open or bridging faults. Other faults are covered only by the general mechanism for bypassing faulty switches. Extra fault protection within each switch could be counterproductive, as it provides more opportunities for faults. Stuck protection is important because it protects fault-free switches from being disabled by configuration requests from faulty neighbors.

Switch addresses set through software could mistakenly be changed by commands from faulty switches. A significant measure of extra confidence is provided by hardwiring addresses into every switch node. Then each configuration command simply specifies its

target address.

Hardwired addresses for each switch can be fabricated with changes to just one layer. This layer can even use step and repeat, if the exposure can choose one of four masks, all of which overlap slightly at the corners. Careful arrangement of these four masks can provide all $4^4 = 256$ possible addresses for an array of 16x16 switches. One such arrangement is shown in Appendix 1 [17].

To send data through the switch network, a path of switches must be configured from the inputs at the wafer bottom to the outputs at the wafer top. First, external configuration lines are raised to initialize the edge switches to receive data from the pads. Then configuration sequences are sent to each edge switch (in parallel if the external bandwidth supports it). This configures the switches next to the edge switches to accept data sent through the edge switches. Then those switches are sent configuration sequences to alert their neighbors, and so forth.

Runtime error correction requires some form of error detection. While it is easy to send patterns through each switch path and check that they are correctly received, it is not so easy to check whether they did indeed pass through the configured switch path, rather than by some other unexpected route. To test whether data is going through the configured switch path, an extra **match** bit is added to each bus. The address matching circuit is augmented to pass this bit through, setting it high if the current command on the switch bus contains the hardwired address for this switch. Then a switch is tested by first sending a null command with no valid switch address (to reset the match bit), and then sending the switch address to check whether the match bit rises. This is repeated for each switch in the configured path, and implemented by a couple of extra gates in the switch address comparator circuit.

3.3 Mask Bits and Request Lines

Each switch node contains 5 mask bits, of which three are for inputs from neighboring switches (south, east, and west), and two for inputs from nearby RAM sites (southwest

and southeast). These correspond to the 5 arrows directed into each switch node in Figure 3.2. The lengthy configuration sequence described in section 3.2 is used only for establishing and testing a switch path. Each configuration line sent to a neighbor must not only set the neighbor mask bit to receive data from the configuring switch, but also reset all the other mask bits so that it will not be competing with data from other possibly faulty switches or sites.

Once a fault-free path has been established, the mask bits for site outputs must then be set, so that each site can be tested. Since the path is known to be fault-free, this can be done with a single load mask command sent directly to the appropriate switch. Figure 3.6 shows the mask bit circuit with 3 configuration inputs passing through stuck detectors, and 5 bus inputs for loading the mask directly. The outputs are delayed one cycle before passing to the request circuit that chooses which fault-free input to receive.

Every data item in the switch network is preceded by a request signal. This gives the switch node one phase to choose which input to accept, and drive the corresponding controls for swH and swV. The request handler circuit is shown in Figure 3.7. Alternatives here include a busy acknowledge line to signal unselected inputs to hold data till the next cycle, or externally settable priority selection. Such controls are not needed for the pipelined memory, however a circuit design for congestion has been presented in [26].

3.4 Switch to Switch Interface

The switch interface contains tristate buffers to route all the bus signals in a common direction. Control for these buffers is generated by the switch nodes on each side of the interface, or just the receiving side for a unidirectional interface. Diagonal cell layout reduces the required circuit area in one dimension to the bus width plus one cell, as shown in Figure 3.8. Figure 3.9 shows the contents of each cell in the horizontal switch interface (swH). These diagonal cells can be spread out appropriately to match the input and output requirements of the sites above and below. This site I/O flexibility and the compressed diagonal layout are possible because of the choice to use cross-shaped switch buses at the site corners. In addition to receiving clocked control from the switch nodes

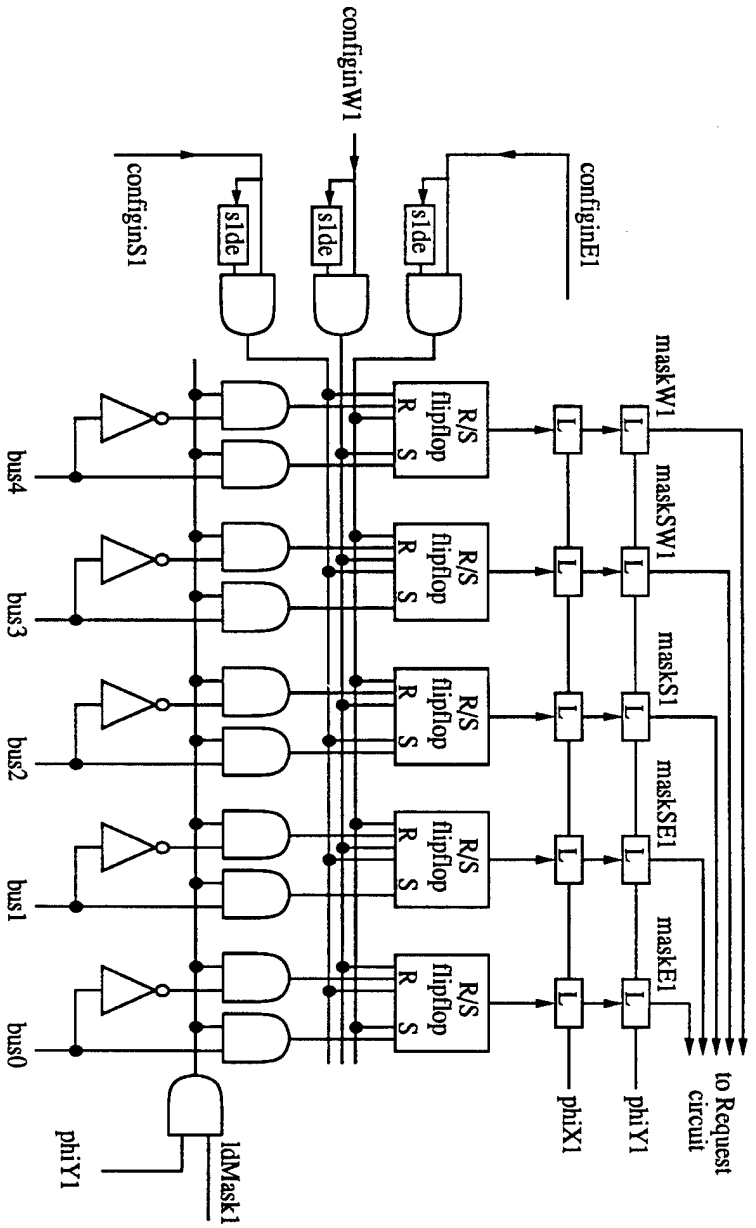


Figure 3.6: Mask Circuit. L boxes are flow-through latches. S1de boxes pull down stuck-high inputs.

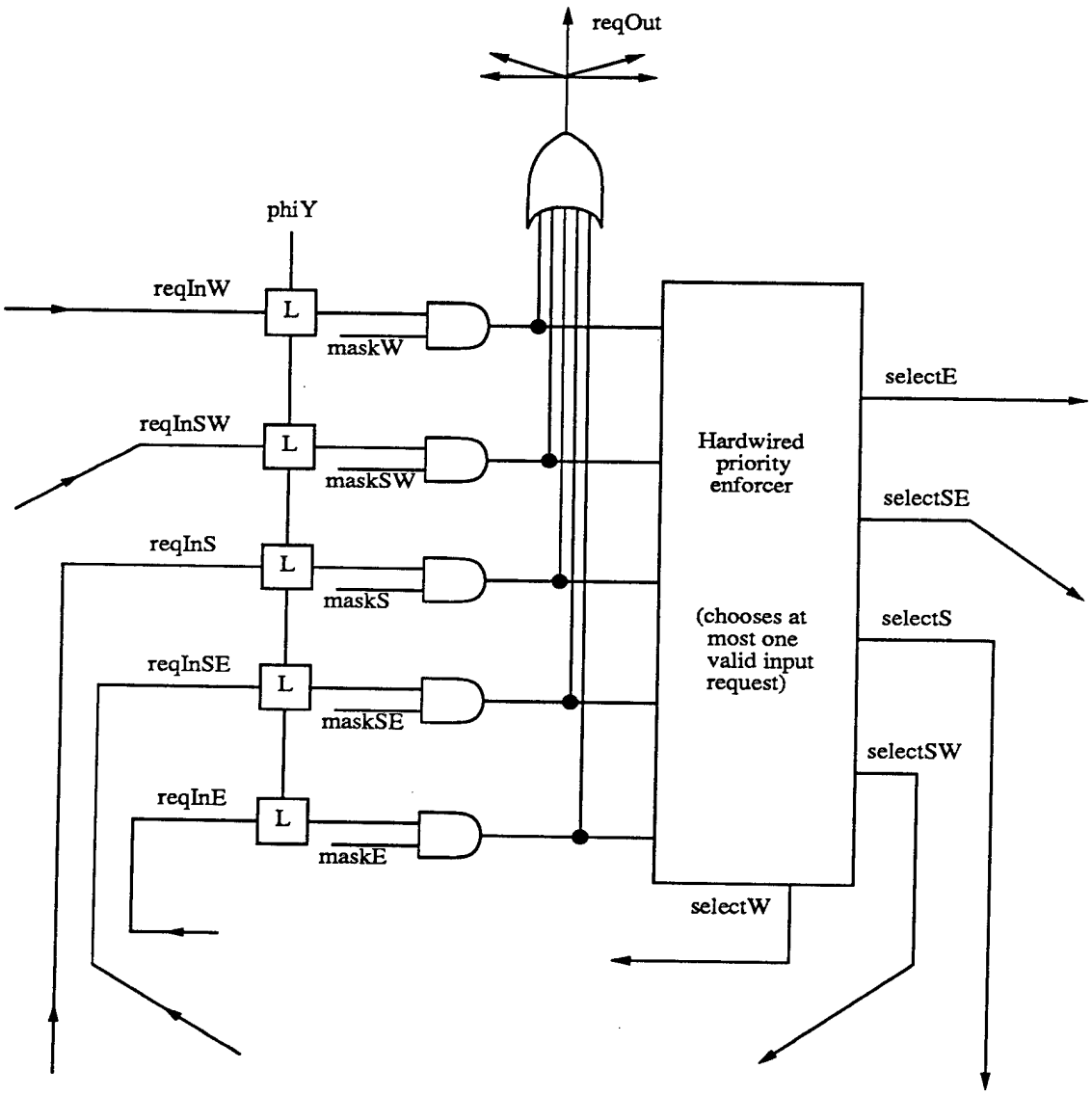


Figure 3.7: Request Circuit. Hardwired priorities for the 5 input directions are implemented with 8 gates. Priority selection guards against external system errors that result in 2 valid data items arriving at the same time.

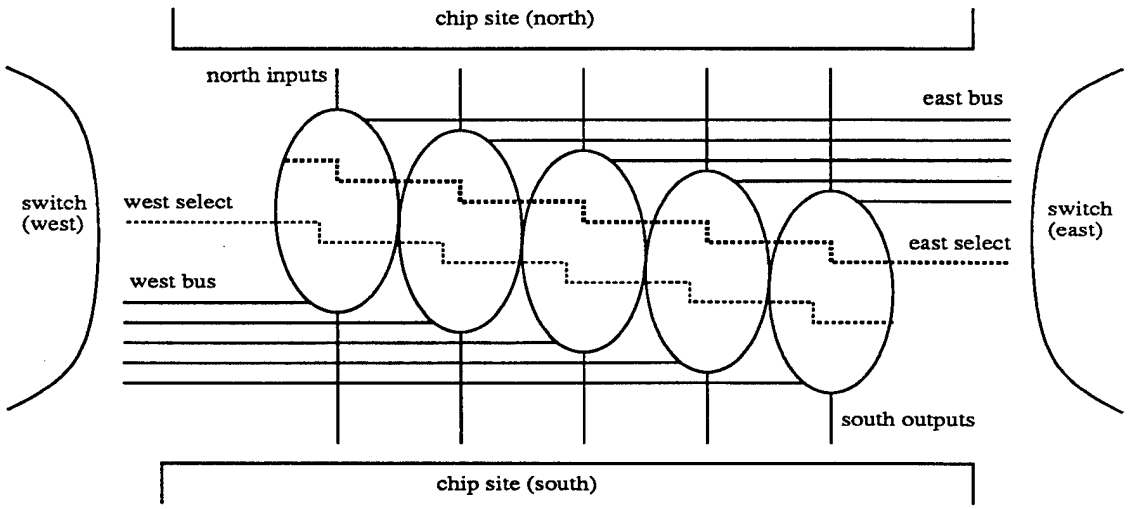


Figure 3.8: Diagonal layout of a horizontal switch interface (swH)

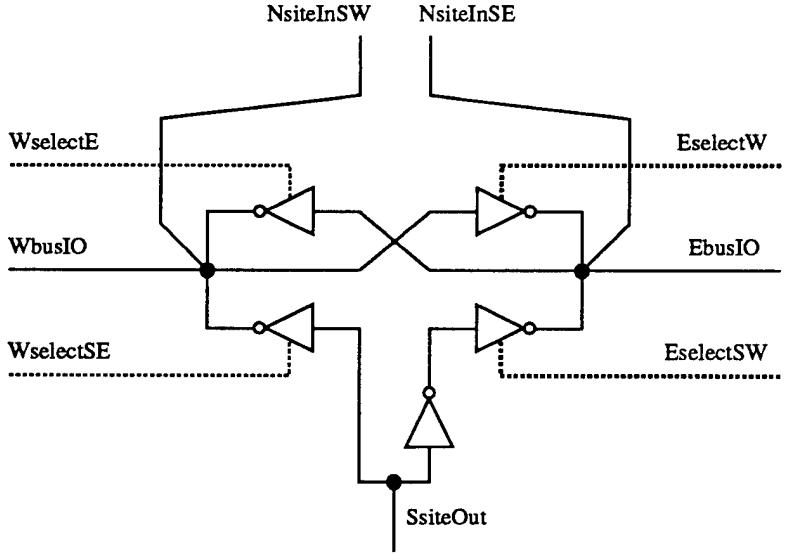


Figure 3.9: Contents of swH cell. The dotted control lines are used to tristate the inverters. The swV cell is simpler, containing only one tristated inverter controlled by NselectS.

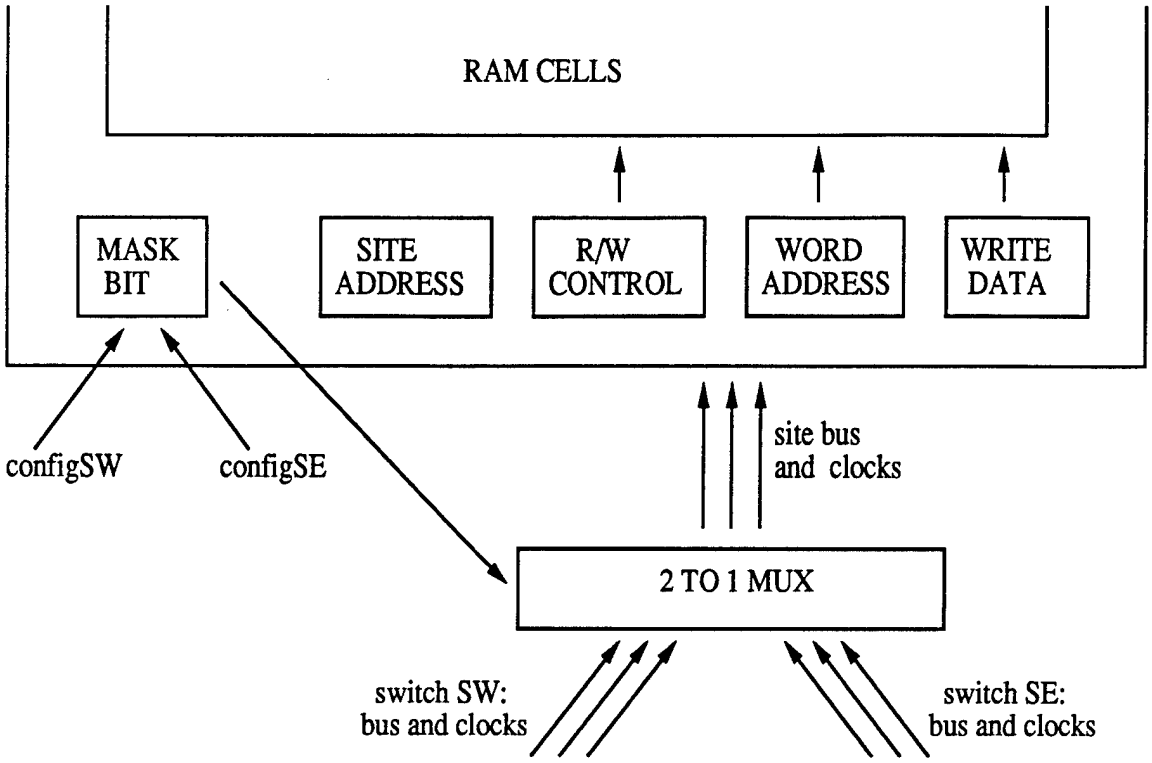


Figure 3.10: Switch-Site interface block diagram.

at each end, the interface cells also require power and ground lines from switches on both sides to be independent of power faults.

3.5 Switch to Site Interface

Each RAM site has one mask bit that chooses whether to accept input from the SW or the SE switch nodes. It also has a soft-settable site address (distinct from the address within the RAM). An outline of the switch-site interface is shown in Figure 3.10.

As with the switch node mask bits, both configuration lines pass through stuck detectors before reaching the mask bit flip-flop. This mask bit controls multiplexors to choose which switch bus to read, and also which request signal, which switch address match and which clock lines. The request and switch address match lines are used to check that the

bus contains valid data.

The RAM site accepts the following commands:

set site address: A new site address is loaded.

write: A 4-cycle write sequence is initiated. The RAM address is latched immediately, and the data is latched on the 4th cycle.

read: A 4-cycle read sequence is initiated. The RAM address is latched immediately, and the data is sent out on the 4th cycle, preceded by a request signal.

set mask bit: Unlike the others, this is not an explicit command on the bus, as the configuration lines from the switch configuration detectors directly set the mask bit.

The bit codings used for these and the switch commands (set mask bits, and configure neighbors) are listed in Appendix A. They use 17 bits, with 16 bit data words for the RAM. Expansion to 32 bits requires either two parcels for each data item, or a wider bus.

The 4-cycle sequences are controlled using a busy bit and a counter. The busy bit is set in the first cycle, cleared in the fourth cycle, and used as an indicator that a command is in progress, so that any otherwise valid commands from the bus are ignored. The counter is implemented as a 4-cycle tag shifter, similar to the 13-cycle counter in the switch configuration detector. It is used to generate the output request signal for read data, latch the write data, generate write enable, and reset the busy bit. These 4-cycle commands can be pipelined, as described in section 3.7.

Each RAM contains 4k 6-transistor static RAM cells. The design was borrowed from the MIPS-X Icache [1]. The RAM outline is shown in Figure 3.11. It is laid out in two blocks of 64x32 cells, with decode at the bottom, and sense amplifiers centered between the blocks. Wordlines are vertical, and bitlines are horizontal. Decode is broken into two steps: a predecode of bit pairs followed by a three-way nor. Writes pull down one of two precharged bitlines, which overpower the cell value. The static RAMs are operated

over 4 cycles to simulate slower dynamic RAM. Static RAM provides the desired site functionality without unnecessarily complicating the design with the fine tuning required for dynamic RAM; this is especially hard to achieve through MOSIS due to variations between different foundry technologies.

3.6 Global Signals

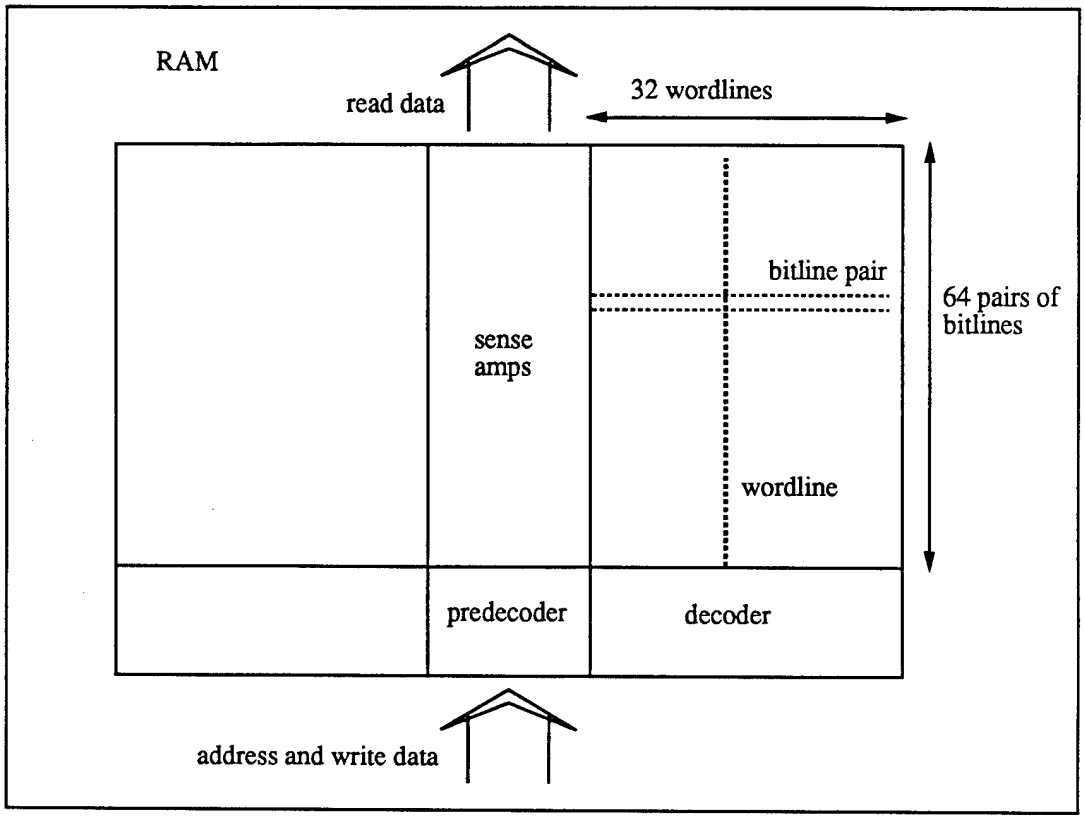
Global signals provide a challenge for every wafer scale system, due to their potential for widespread damage from a single fault. Power and clock lines, the only global signals in this implementation, each have special characteristics that affect their layout.

3.6.1 Power Lines

Power nets must be very wide, unless special (more expensive) packaging allows bonding to the center of the wafer. Wide power nets reduce the resistive drop to a safe level (under 0.5V) to retain acceptable noise margins on all internal signals. Thus, a resistive soft-configurable transistor switch is not practical for switching power lines.

Several WSI researchers, including the RVLSI group at Lincoln Labs, have documented the need for internal filter capacitors for very long power line distribution. Otherwise noisy power lines (usual in CMOS) can lead to intermittent signal errors. Power resistance and capacitor design are examples of typical board design issues that become on-chip issues in WSI.

Peak power for the RAM is around 117 mA for GND, and 78 mA for Vdd. Using a rule of thumb of 1μm per mA requires power lines considerably wider than the switch layout even for small numbers of RAM sites. As an example, consider a system of 6 RAMs per row, with power fed from both sides. The outer sites are fed directly, but the inner sites (2 per side) require around 400μm width just for site power. When spacing and switch power is added, this is already wider than the switch circuit. Space can be saved by running power over the switch in metal2, however with standard bonding technology



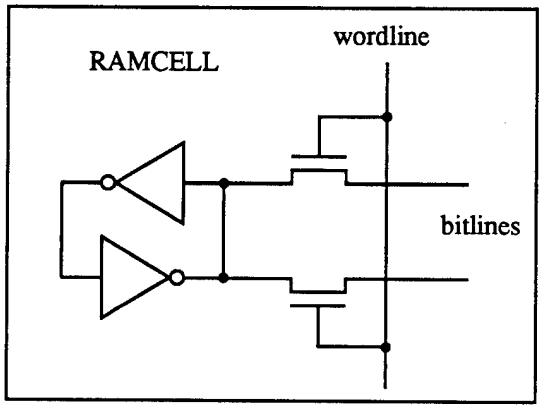


Figure 3.11: Ram Outline

the power lines and not the switch will dominate space requirements. This space saving was used in the implementation; however it is not free. Running power over the switch circuitry increases the probability of power line shorts.

For the small example implemented (2x2 sites), power line width was not a problem. Tolerance of power line shorts was provided by separating the power net. With a separate Vdd for each site and switch, power shorts are contained to affect only the switches or sites containing them. This adds testing overhead. In addition to the use of DIP switches to isolate each power net from the test power supply, extra tests are needed to use these DIP switches in the event of a short disabling part of the circuit.

Care is required in circuit design to retain separation of the power nets. The horizontal switch interface needs two sets of power lines, so that a short on one switch will not affect its neighbor. Also, every multiplexor in the switch-to-switch and switch-site interfaces must be implemented using tristate drivers, rather than simple transmission gates. Inconsistencies in the power separation of the first fabrication run were traced to the use of transmission gates in the switch-site interface. When a site is powered down, the transmission gates pass signals from one switch bus directly onto the neighboring switch bus through the switch-site interface.

Power net distribution could benefit greatly from several innovations in technology. These include low temperature operation, extra thick metal lines, extra metal layers, bonding directly to the center, and laser fuses or antifuses. However, each innovation has costs, detracting from the low-cost goal of soft configuration.

Power lines can be switched using very wide transistors that spread over considerable quantities of silicon. Such devices as the Built-In Current (BIC) sensor are described by Wojciech Maly [31, page 49]. The BIC senses whether the current is high; if so it activates a circuit breaker that disconnects the local power lines from the global power bus. This catches many common CMOS defects, including power to ground shorts. Such devices are definitely worth consideration, however may not always be practical; a rough estimate of the RAM site power needs required a power transistor with width to length ratio exceeding 2800.

3.6.2 Clocking

Clock lines can be switched, subject to skewing constraints, greatly easing the task of providing fault tolerance in comparison to power lines. However, soft configuration of clocks would require an asynchronous design, or recursively more clocks to clock the clock configuration. A simpler solution can be implemented using dynamic fault masking.

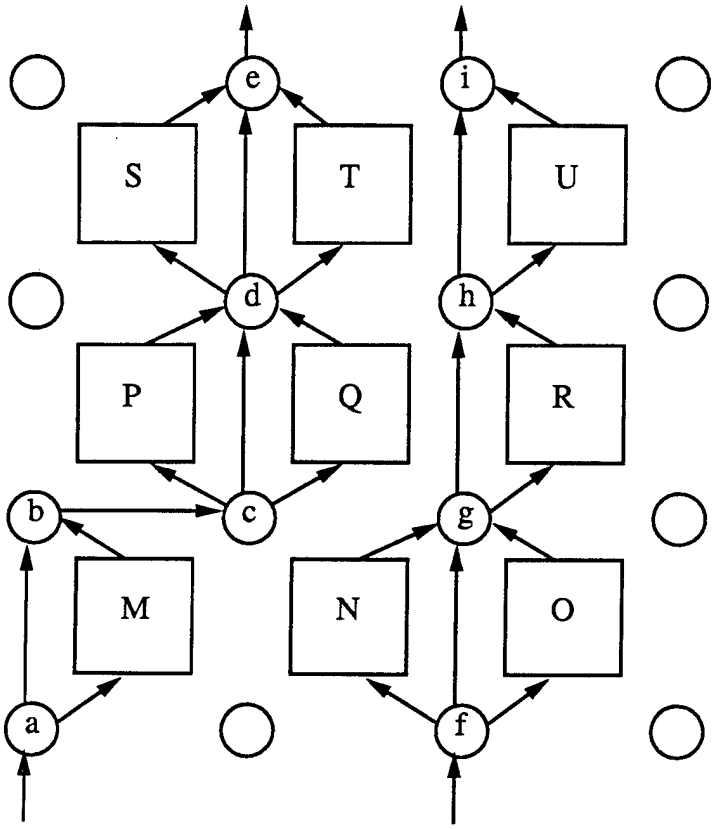
The clocking circuit shown in Figure 3.3 has four inputs and two outputs. The four inputs are ϕV1 and ϕV2 (fed vertically from the top or bottom of the wafer) and ϕH1 and ϕH2 (fed horizontally from the left or right). Each line goes through a large stuck-detector, as in Figure 3.5. These are designed to switch in 17μs, leaving ample time for every line to switch in a system designed to have 25ns phases (20MHz). The vertical and horizontal clocks are then OR'ed together, providing switch clocks (called ϕX and ϕY) that work even if one of their inputs is stuck high or low. Skew between the vertical and horizontal clocks results in a switch clock that is high as long as either input is high, requiring a longer downtime between phases to avoid overlap.

Switch path latency can be halved by using alternate clocks on neighboring switches. If a switch receives data in phase 1, then its neighbor switches receive the data in phase 2, which is their phase 1. Switch clocks are thus arranged in a checkerboard, where the white squares have (ϕX = ϕ1, ϕY = ϕ2) and the black squares have (ϕX = ϕ2, ϕY = ϕ1). A site must use the same clocks as the switch it reads, since data will arrive on its input switches during opposite phases.

One drawback to checkerboard clocking is the need to balance skew very carefully. Another drawback is that each phase must last the same amount of time, eliminating design styles with one phase shorter than the other.

3.7 Configuration and Pipelining Example

Figure 3.12 shows the steps required to configure two example switch paths. The left path (a-b-c-d-e) contains a jog, assuming the input switch between a and f is faulty. The



1. Wake (a)	1	Wake (f)	1
2. Config a to wake (b,M)	14	Config f to wake (g,N,O)	14
3. Config b to wake (c)	14.5	Config g to wake (h,R)	14.5
4. Config c to wake (d,P,Q)	15	Config h to wake (i,U)	15
5. Config d to wake (e,S,T)	15.5		
6. Setmask e to also read (S,T)	3.5	Setmask i to also read (U)	2.5
7. Setmask d to also read (P,Q)	3	Setmask h to also read (R)	2
8. Setmask b to also read (M)	1.5	Setmask g to also read (N,O)	1.5

Figure 3.12: Example system showing simultaneous configuration of two paths with cycle times for each step. In step 1, external configuration lines are pulled directly. Steps 2-5 are strictly sequential, as each configuration relies on the previous one. Steps 6-8 are independent, so may be overlapped. If steps 6-8 are initiated one cycle apart, then this group of requests will be completed inside 4 cycles. The half-cycle times reflect the bus delay of 2 switch nodes per cycle, due to the clocking described in the previous section.

configuration shown provides access to every site. The first group of commands (steps 1–5) shows configuration of both paths at once. These configurations must be strictly sequential, with each step completing before the initiation of the next. The second group (steps 6–8) lists commands that set the switch masks to receive site outputs. These can be pipelined by sending commands 6, 7 and 8 in three consecutive cycles.

Two command groups have been omitted here. One is the testing of switch paths using the match bit (after configuration but before setting masks for site outputs, so between steps 5 and 6). The other is the commands that set site addresses. Site addresses must be distinct within each path (M,P,Q,S,T and N,O,R,U), unless memory duplication is needed for runtime fault tolerance. Path testing and site address commands can be pipelined in a similar fashion to steps 6–8.

Figure 3.13 shows a sequence of steps that write data to the sites. Note that the write pipeline must be broken after 3 writes in order to send in the data to be written. Figure 3.14 shows a sequence of reads. Reads can be sent through continuously, provided that data from one site does not squash commands addressed to another.

3.8 Implementation Experiences

I borrowed the static RAM design from the MIPS-X on-chip cache, planning for a quick implementation, so that more of the design time could be spent on the switch network. Unfortunately, a series of mishaps (described below) led to serious problems in the static RAM, leading to the RAM taking by far the largest share of testing time to diagnose and correct.

We originally planned to build 2x6 sites, however settled on 2x2 so that it would fit inside a regular package for testing. The 144 PGA package available could not fit all 168 pins required for 6 switch buses (3 inputs and 3 outputs), with every clock and power signal separate. So two of the output buses were multiplexed, and some of the clock lines were joined. These choices left every power signal separate, while retaining redundancy for clock failures. The design interface includes 3 separate clock lines (both phases)

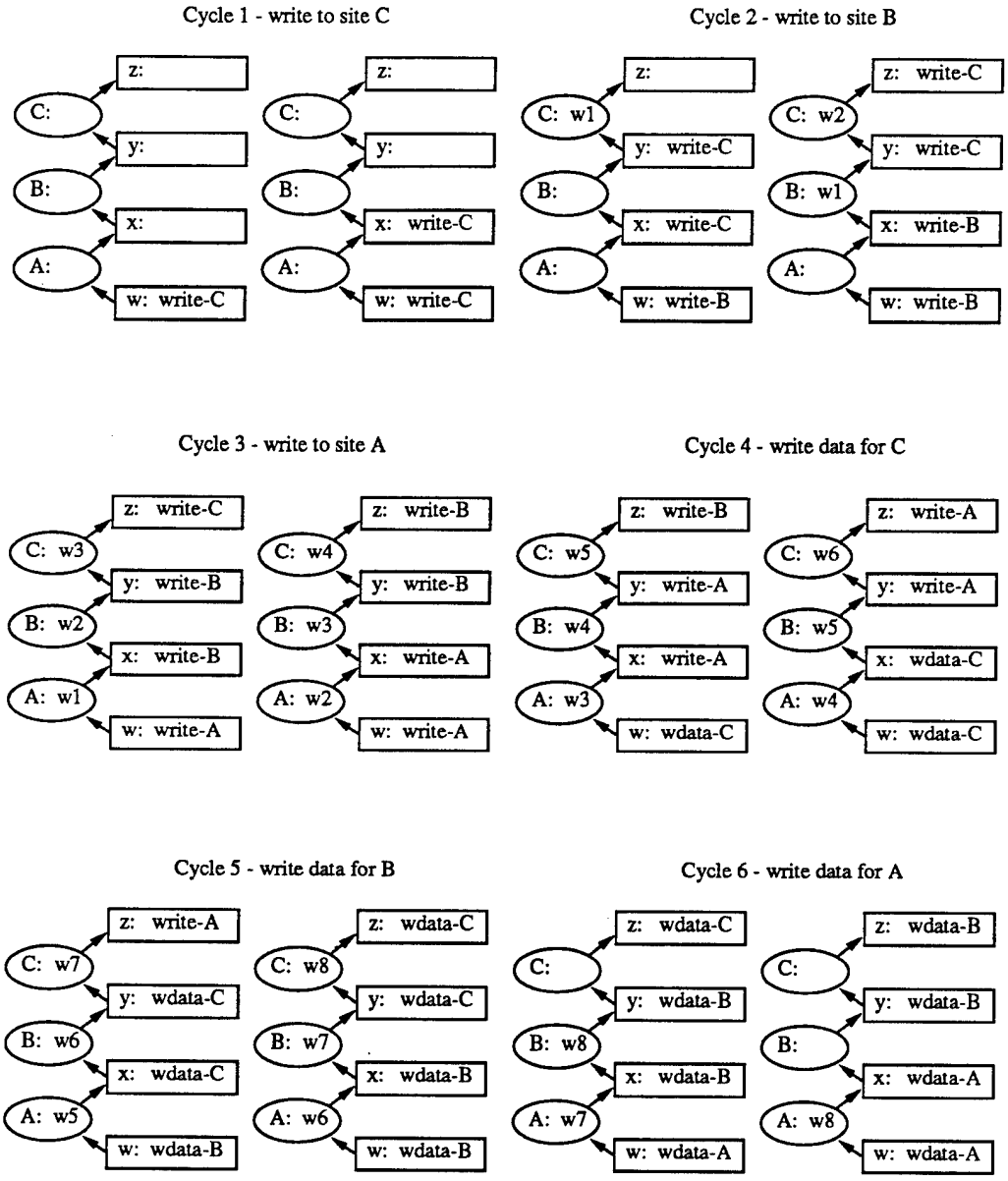


Figure 3.13: Write sequence. Addresses are latched in cycle 1, write data is latched in cycle 4. Although 4 cycles are needed for each write, all 3 writes complete in 6 cycles, due to pipelining. Switches w,x,y,z are shown with their bus contents. Data enters switch w, and progresses up through x, y and z. Sites A,B,C are shown with the phase (1–8) of their 4 cycle operation.

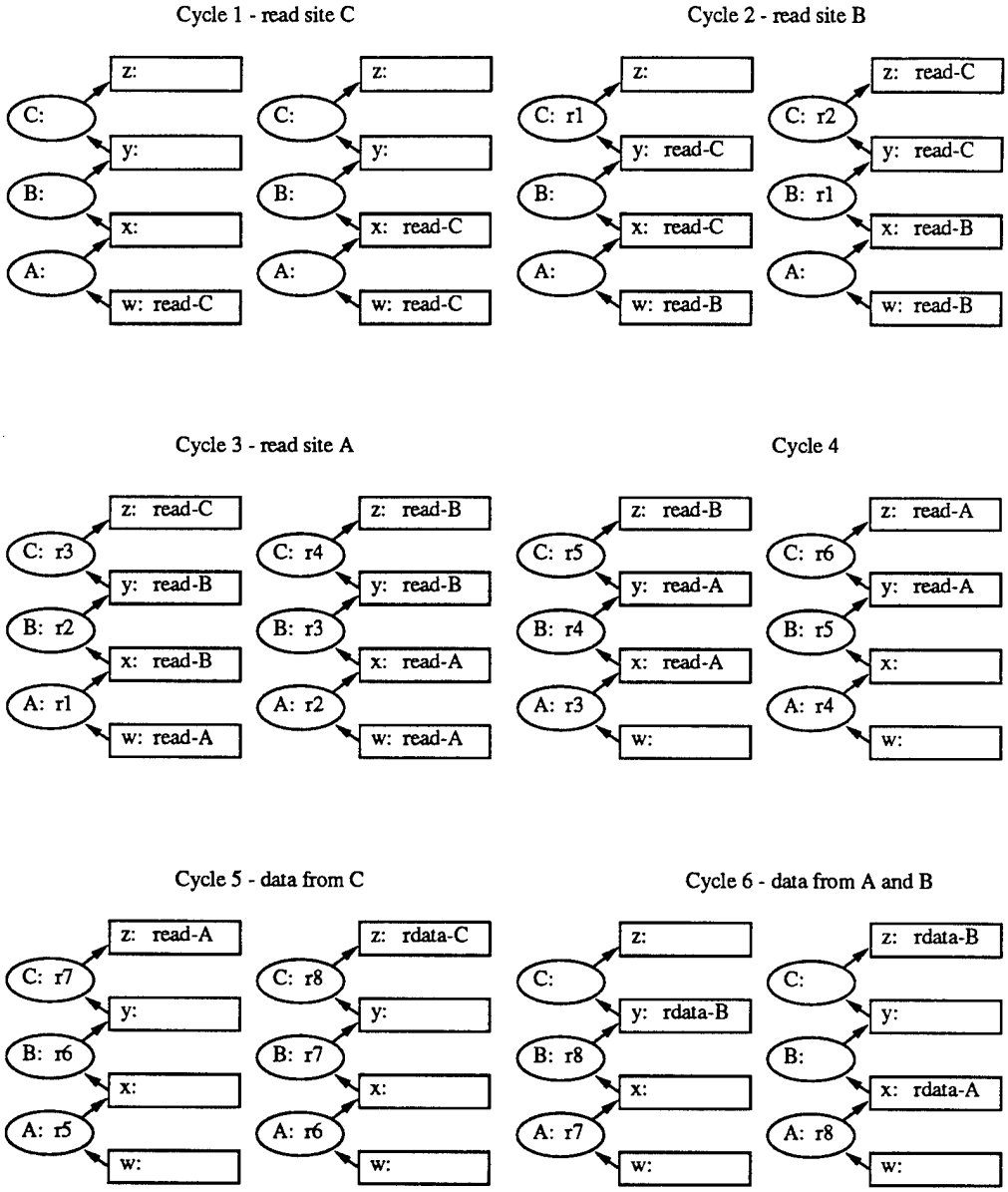


Figure 3.14: Read sequence. Addresses are latched in cycle 1, site data emerges in the second phase of cycle 4. Although 4 cycles are needed for each read, all 3 reads complete within 6 cycles, due to pipelining. A 7th cycle (not shown) would bring site A data up to output switch z. Switches and sites are shown as in Figure 3.13.

entering from top, bottom, left and right. Two from each side were joined, requiring a total of 16 rather than 24 clock pins.

The request inputs were tied high. This saves 3 pins (1 per input switch), but restricts switch paths to be linear, without forks and joins for two-sided bypasses. Bypassing on two sides provides access to both sites near the faulty switch from the one switch path. The fixed routing priorities favor one switch input over another, irrespective of where the data came from. Clearing the external request line leaves the choice open to whichever switch input holds data generated by a site on one side of the fork.

These pin savings reduced the total to 142 pins, leaving 2 for testing. One was used to bring out a switch clock, to verify that the clock was working correctly through the stuck detection circuitry. The other brought out a switch mask bit, to check that it could be correctly configured.

The switch implementation was fairly straightforward, after some attention to inserting the right number of inverters for the checkerboard arrangement in which half the switch buses were active low. The switch used relatively few transistors. Figure 3.15 shows the breakdown of area and transistors for each part of the design. The transistor switch total includes all the pieces listed below, including the switch to switch interfaces.

The configuration detector and horizontal switch interface were the largest components, with 415 and 316 transistors respectively. The horizontal switch interface is much larger than the vertical interface because it is bidirectional, and includes site input and output wires. The area figures are for 2μm CMOS. With 4 RAM sites, and 9 switches, and including interfaces, this comes to around 32.6mm^2 RAM area and 17.3mm^2 switch area. The switches take around a 1/3 of the active area (excluding pad wiring). This ratio can be improved by using larger sites with internal redundancy, as switches are relatively independent of site size. Note that overhead due to switch area should be compared with the area needed to package chips separately, or in a multiple chip carrier, or alternative redundancy schemes. Comparison against a system with no switches is meaningless, as such a design would have zero yield in a wafer implementation.

Magic and RSIM (both distributed with the Berkeley CAD tools, and modified locally)

Component	Area($mm^2 = 10^6\lambda^2$)
switch control	1.4
sw/sw vertical	0.22
sw/sw horizontal	0.56
sw/RAM interface	0.56
RAM	7.6

Component	Transistors
RAM	28,000
switch total	1266
sw clock	72
sw decode	50
mask	151
request	136
address match	52
config detect	415
swH	316
swV	74

Figure 3.15: Switch and RAM area and transistor breakdown

were used for layout and simulation. Simulation scripts were generated using a simple assembler written to translate reads, writes and configuration requests into the appropriate sets of 1's and 0's. This eliminated the error-prone task of hand-rotating the configuration sequences, and deciding which ones should be inverted because their source or destination switch is.

Testing required separate power and clock signals for the test board, so that fault tolerance on the die was not sabotaged externally. This was provided by inserting DIP switches between the tester signals and the chip signals. Resistors were included so that the chip signals would be pulled to a known value (ground) when the DIP switch was open. However, splitting one tester signal into many chip signals had the effect of putting all the resistors in parallel when the DIP switches were closed. This reduces the effective resistance, however caused no problems with the 1 Mohm resistors used. The ground net was fully connected, only the power net was separated, and with 19 resistors in parallel, the resistance dropped to around 52 Kohm.

The tests were translated directly from the simulation scripts using awk. This helped to eliminate bugs in the tests, as they were always checked first against the simulated results. The assembler was modified to generate print requests for the simulator that could be translated directly to tester requests to examine the output pins. Tests were generated to verify every switch connection, so that switch faults could be located.

Testing uncovered several problems, most of them minor. The stuck detectors only kept the clock signals high for around $2\mu s$, rather than around $18\mu s$ as predicted by simulation. This was due to miswiring the capacitor, and was fixed in a later run. It had no ill effects, as $2\mu s$ was easily long enough for everything to switch, and $18\mu s$ was not really needed (however there was plenty of space for large capacitors due to other layout constraints).

The clock redundancy worked correctly. This was verified by opening a DIP switch on one of the clock inputs, and checking that the other one was still driving the switch clock with no effects apparent on the test output clock pin.

The mask bit available for testing at first appeared not to work. However this was

traced to a bonding error, as one of the input pins was not bonded. Thus the first real use of the on-chip fault tolerance was to reroute the data from another set of input pins to test this mask bit.

Rather than individually test every DIP switch, they were split into groups corresponding to the left, middle and right switch columns. Power separation worked for the switch to switch interfaces, but there were some oversights (transmission gates rather than inverters) in the switch to site interface. These transmission gates transferred data directly from one switch to its horizontal neighbors through the switch to site interface when site power was disabled. So power separation worked only when the nearby sites were enabled. This problem was corrected in a later run.

The switches functioned exactly as designed, however there were serious problems with the RAM. Reads appeared to work correctly, however writes could only switch bits from 1 to 0, not the other way. This was diagnosed by probing the $10\mu m^2$ probe points included in several critical places. The problem resulted from a critical race in the MIPS-X cache design. This undocumented feature was not caught by the simulator due to some outdated capacitance values in the Magic technology file. The slightly incorrect node capacitances were enough to allow the simulator to win the critical race by around 3 ns. It was solved by qualifying the Write Enable signal with $\overline{\phi 1}$ rather than $\phi 2$.

The critical race problem has been solved in a later run, so writes work reliably, however a new problem with read timing has not been diagnosed. Site RAMS sometimes send read data out to the switch network one cycle early. Testing was hindered by an operating system update that outdated the kernel driver for the Medium tester, for which no working source is available.

The testing proved the switch network to be 100% functional, with both power separation and clock stuck-detection working correctly. Site RAMs now function correctly, but there appears to be a problem with the switch to site interface related to timing of the read data. Speed has not been tested.

Chapter 4

Circuit Unit Yields

Fault-tolerant circuit designs clearly have some effect on system yield. What that effect is can only be determined by evaluating the yield of every circuit block, and then factoring in the effects of fault tolerance to produce a combined global yield. In the next chapter, block yields are combined to estimate the global yield of the pipelined memory. This chapter describes how to obtain the block yields that provide the basis for this.

Block yields are estimated using Monte Carlo fault simulation, where defects are added randomly according to input distributions, and each iteration checks whether the added defects changed the circuit. Defect measurements are compared to fault simulator results for the same test patterns.

4.1 Poisson Yield Models

The most accurate way to assess yield is to build thousands of wafers, and test them exhaustively. This method has several drawbacks. One is that it is exceedingly expensive; if yield is found to be insufficient, there may not be enough time or money left to try alternatives. Another is that it gives results only for those fabrication lines actually used, and is thus dependent on particular fabrication machines and methods. If the industry continues to move as fast as in the past, then by the time results are available, they have

been outdated by newer and better machines and methods.

The only alternative is to predict yield, based on results from previous fabrications using similar processes and equipment. The model most widely used by the industry is also the oldest and simplest. If defects are random and uniformly distributed, then straightforward mathematical manipulations produce the Poisson Yield Model.

Suppose a site has area A. Assume that for each small dot $\triangle A$, the probability that it contains a defect is $\alpha \triangle A$, where α is a constant that depends on the quality of the fabrication line, and $\triangle A$ is sufficiently small so that the probability of it containing two or more defects is negligible. A binomial distribution can be used to estimate the probability that a site of area A contains i defects. To do this, set $n = A/\triangle A$ which is the number of these small units of area, and regard the existence of a defect in each small unit as an independent trial, with each trial having a constant probability $\alpha \triangle A$ of containing a defect.

Then the probability of finding i defects in an area $A = n \triangle A$ is

$$p(i) = \binom{n}{i} (\alpha \triangle A)^i (1 - \alpha \triangle A)^{n-i}$$

Substituting A/n for α leads to

$$p(i) = \frac{\left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right) \cdots \left(1 - \frac{i-1}{n}\right)}{i!} (\alpha A)^i \left(1 - \frac{\alpha A}{n}\right)^{n-i}$$

As n increases this approaches the Poisson distribution

$$p(i) = e^{-\alpha A} \frac{(\alpha A)^i}{i!}$$

that for defect-free sites ($i = 0$) reduces to the yield probability

$$p(0) = e^{-\alpha A}$$

The approximation is valid when n is large, and αA is small. The derivation is described in many standard books on probability theory (such as [34], p49) and applied specifically to yield in [38].

Fabrication consists of several steps, each with its own defect rate. If these rates (α_j) are independent, yield can be calculated by multiplying the probability of 0 defects for each step:

$$\text{Yield} = \prod_j p_j(0) = e^{-A\sum_j \alpha_j}$$

Independence of fabrication steps is one of several assumptions known to be imperfect. Another is that yield is limited by random defects. Other sources of failure include faulty equipment, design errors, and process parameter problems.

Modelling defects using a binomial distribution requires another independence assumption: that the probability of a defect in each small unit of area is independent of that in any other area. This assumption has been repeatedly challenged over the years, based on measurements and on known features of IC processing — for example a misaligned mask will affect the entire die in a similar fashion. Higher than expected yields can be explained by assuming clustering, where defects are bunched together on a smaller number of dies than would be expected in a uniform distribution. However, recent carefully controlled tests [30] have failed to substantiate the existence of a substantial clustering component.

Yet another assumption is that defects are directly proportional to site area. Since parts of a site can be featureless, a common modification is to instead use a parameter called **critical area**, which is the area that does contain circuits.

The term **defect** is used here to mean the actual physical modification of some part of the fabricated circuit. Only in some locations would such a defect actually change the circuit, in which case we call it a **fault**. This is because the defect could be so small as to make no substantial change to any wire, or in a location that is empty of wires.

These rough approximations, using area, critical area, or number of transistors, do not provide a high level of confidence when applied to a particular circuit design. However, the same formulas can be applied using full knowledge of the circuit layout, as is done in simulators like VLASIC.

4.2 Introduction to VLASIC

VLASIC (VLSI LAyout Simulation for Integrated Circuits) is a Monte Carlo yield simulator [45]. It uses the circuit layout, process technology parameters, and defect statistics to predict yield. Defect statistics are used to generate random sized defects conforming to a defect distribution that was observed in the past, or is postulated for the future. The process technology defines how the defects affect the layout.

Generated defects are added to random places in the circuit layout, and an analysis phase decides whether the defect changed the circuit configuration. If so, then that defect caused a fault. The process of adding randomly sized defects to random places in the circuit is repeated many times to obtain an average that predicts the yield at fabrication. For instance, if faults occur in 50 of 10000 trials, it predicts a yield of 99.5%.

The inputs described here are specific to VLASIC 1.1. Later versions are incompatible, using the same information specified differently. The principal change is in the technology description, where version 1.1 is limited to exactly 8 fault types (short circuit, intralayer open, open via, new via, new gate device, new active device, open device and shorted device). These limitations are not greatly significant for here, as errors in the defect statistics are likely to overwhelm any minor discrepancy involving fault descriptions. Despite such problems, the results are expected to provide a reasonable estimate not of absolute yield, but of the relative yields of the switch circuit and the RAM circuit. While absolute yield predictions are tenuous, they remain the only alternative to constructing large numbers of wafers and testing them all.

VLASIC is not limited to the simple Poisson distribution; given appropriate inputs it will provide negative binomial clustering [44] within each wafer, across wafers in a lot, and across lots. Clustering is assumed to explain higher than expected yield, where multiple defects group together on a small number of ill-fated sites, leaving fewer defects for the remaining sites. Clustering could have a negative effect at the block level — if fault-tolerant spares were only located close by, then one clustered fault could wipe a circuit unit together with all its spares. This possibility is of particular concern with heterogeneous systems containing many different cell types such as the

Trilogy computer design. Placing the spares far apart has drawbacks in performance, as the longer communication wires incur a higher resistive load. However, clustering has usually been observed at a smaller level than an entire circuit unit — on a scale of 10s of microns rather than millimeters. Following the experimental results of Lukaszek[30], the data below was prepared without clustering parameters, assuming uniform distributions throughout.

VLASIC provides one more spatial variation in defect placement. Two defect zones may be specified, to model the effects of handling which adds extra defects to the wafer rim. For a rectangular design such as the pipelined RAM, only the corner sites are near the wafer edge. No defect zones were specified for the data below.

Each defect must be assigned a size. Defects smaller than a minimum diameter determined by photolithographic limits will have no effect on the circuit. VLASIC provides a continuous size distribution taken from the literature: below the diameter defects are assumed to fall off linearly, and above it they fall off as $1/x^3$. The defect diameter is the most common defect size. Oxide pinholes and junction leakage are always assumed to have 0 diameter. The continuous defect size distribution overestimates yield, since measurements of wire test pattern shorts show that defect sizes fall off considerably slower than $1/x^3$ [30]. These wire test pattern yields were used to specify the defect size distribution as a collection of discrete ranges, where defects occur uniformly within each size range, and frequencies are specified for each range. This distribution is the average of 3 measured ones, and is valid only for small defect sizes, as the experiment was unable to determine the the size of defects shorting more than 4 wires. Figure 4.1 shows the size ranges and frequencies for this average distribution. The continuous distribution was used for defects other than shorts, for which no size data was available.

Finally, VLASIC requires the defect densities for every defect type specified in the technology file. Recall that this is the frequency with which defects appear on the silicon, not the (lower) number of defects that actually cause faults. For shorts and opens, these frequencies are measured by fabricating dense interlocking wire patterns, where every defect above the minimum size causes a detectable fault. Experimental yields of wire test patterns show that shorts occur about 10 times more frequently than opens[30]. A

Size Range (μm)	Frequency
1.25 - 4.5	0.623
4.5 - 7.75	0.241
7.75 - 11	0.137

Figure 4.1: Defect size distribution. VLASIC uses this data to generate defects with diameters uniformly spread within each size range, where a size range is chosen according to the specified frequency. No defects are generated smaller than 1.25 μm or larger than 11 μm.

figure of 2 shorts/cm^2 was chosen for 2μm spacing, with other defect rates for shorts scaled from this figure according to the square of the spacing. These defect densities, including estimates for missing vias and interlayer pinholes, are shown in Figure 4.2.

4.3 Ram and Switch Yield

VLASIC yield estimates were obtained for the soft-configurable switch and RAM circuits, as shown in Figure 4.3. The switch circuit includes the switch to switch interfaces, and the RAM circuit includes the switch to RAM interface. The first two lines predict yield for the switch and static RAM sites that have been fabricated. The bottom two lines estimate the yield for larger memory sizes by scaling up results both for the RAM cells and for the support circuitry (decoders and sense amplifiers). The larger memory yields are optimistic because power line width, driver size and decoder complexity were not scaled; the same layout is used throughout.

A production design might use denser dynamic RAM, and include spare rows and columns, which could change the yield considerably. The purpose here is not to build the ideal site, but to evaluate the fault tolerance of the switch network, for which the contents of the site are regarded as a black box with a given yield. An equally important purpose is to show that the yield of each circuit unit can and should be estimated and used to make more realistic predictions of the defect tolerance of wafer scale designs.

If this process predicts correctly, then switches will have very high yield. Even

Defect Type	Width/Spacing	Defects/cm^2
Extra polysilicon	2	2
Missing polysilicon	2	0.2
Extra diffusion	3	0.9
Missing diffusion	3	0.09
Extra metal1	3	0.9
Missing metal1	3	0.09
Extra metal2	4	0.5
Missing metal2	4	0.05
Missing poly-metal1 cut		0.1
Missing diff-metal1 cut		0.1
Missing metal1-metal2 via		0.1
Poly-metal1 pinhole		0.4
Metal1-metal2 pinhole		0.4

Figure 4.2: Defect densities per square centimeter for 2 μm CMOS. For "extra" material, line spacing is shown, and for "missing" material line width is shown, in units of $\lambda = 1\mu$m.

Circuit	Transistors	Yield(%)
switch	1266	99.2
4K RAM	28K	93.3
16K RAM	104K	78.8
64K RAM	403K	40.9

Figure 4.3: Circuit Unit Yields

so, when many switches are needed, yield can still be a problem, especially given the uncertainty of this prediction. Four MOSIS runs through different fabrication lines did not give much cause for optimism: one run failed entirely, and others had varying success, none with more than 60% of parts having all 9 switches functioning. A larger proportion (26/30, 18/23 and 25/30) had many or most switches responding to tests. Perhaps shorts between bus lines (where they cross) or from the power lines running across the switches reduced yield further than was predicted by VLASIC. Dependence on switch yield is examined in the following chapter.

100,000 VLASIC rounds of adding defects and checking for faults were used for these yield estimates. For each run, the number of fault-free rounds is divided by 100,000 to produce the yield. VLASIC has been designed to analyze circuits at a low level, giving detailed information on the kinds of faults to be expected from specific layouts. This information can be used to tune design-rules for new technologies. It can also be used to tune a specific design for high yield.

VLASIC breaks the circuit into polygons (permitting arbitrary angles, not just Manhattan design), storing some 2 Kbytes per polygon. This requires plenty of virtual memory, more than was available when these numbers were obtained. So each circuit was first partitioned into smaller pieces, and the resulting yields multiplied, once again assuming spatial independence.

The switch was broken into 3 pieces, and the RAM into 5 pieces. The RAM pieces were

- 2 lots of random logic, including the switch-RAM interface
- decoders and wordline drivers
- sense amplifiers and data bus
- 128 RAM cells

The decoders, sense amplifiers and RAM cells occur multiple times, so the predicted yield is raised to the appropriate power. The 128 RAM cells are weighted by far the heaviest, repeating 32 times for 4 Kbits.

In scaling up to larger sizes, width and height were assumed to scale equally, thus increasing the number of decoders at roughly the same rate as the number of sense amplifiers. An unequal scaling would result in a yield dependent far more on the decoder layout than the sense amplifier layout (or vice versa); however both of these are relatively small contributions compared to the RAM cell layout.

A random selection of static RAM die photos over the last 15 years shows no relation between the size of the RAM and the proportion of area devoted to overhead (decoders, drivers and sense amplifiers). Thus it is not reasonable to assume that the area proportion devoted to overhead drops as RAM size increases. Unfortunately, using the same decoder and sense amplifier cells, even when duplicated an appropriate number of times, results in an assumption that overheads drop substantially as RAM size increases. Maintaining the same overhead rate for larger RAMs requires doubling the size of each decoder and amplifier cell, not just increasing the number of them. Without this, yield estimates for the larger RAMs are somewhat optimistic; a small effect since yield is mostly determined by the RAM cells.

4.4 Comparing VLASIC to Test Pattern Yields

Test pattern yields provided the approximate defect statistics used by VLASIC. I simulated the test patterns using VLASIC, producing the yields shown in Figure 4.4. The test patterns are snakes of 5 parallel wires winding back and forth with width 4mm and height 1.27mm for the 1x structure. The 2x structure is twice the height, and 4x structure is four times the height. Wires are 2μm wide with 1.25μm spacing. The 1x and 2x structure produced the yields shown, and the 4x results are obtained by squaring the 2x results. Squaring is quite accurate using Poisson distributions without clustering: squaring the 1x results produced identical numbers for 8 of the 9 simulated 2x results, with the 9th differing only 0.1%.

The Poisson yield formula,

$$Y = Y_0 e^{-D_0 A}$$

Fabrication Lab	1x	2x	4x	shorts/cm^2
Fab A	97.8	95.7	91.6	1.2
Fab B	93.6	87.7	76.8	3.5
Fab C	95.4	90.9	82.7	2.6

Figure 4.4: VLASIC test pattern yields (%). Results shown use the defect rate and size distributions measured in 3 fabrication laboratories referred to as A,B and C.

Fabrication Lab	Wafer Area	1x	2x	4x	Y_0	D_0
Fab A	wafer	89	86	77	0.94	1.317
	6 x 6	92	88	80	0.959	1.179
	4 x 4	93	89	82	0.97	1.197
Fab B	wafer	82	69	53	0.946	3.943
	6 x 6	86	74	59	0.977	3.493
	4 x 4	91	79	68	1.089	5.736
Fab C	wafer	85	78	64	0.947	2.726
	6 x 6	86	80	65	0.955	2.598
	4 x 4	86	80	66	0.988	3.591

Figure 4.5: Measured defect rates. The *wafer* yields include all 52 dies measured on each wafer. The 6 x 6 numbers include only the center 36 dies, and the 4 x 4 numbers use the center 16 dies.

predicts that with a defect rate of D_0, a circuit of area A will have yield Y, where Y_0 is a factor reflecting systematic defects. The values D_0 and Y_0 and the measured die yields are shown in Figure 4.5. The "wafer" lines give yields for every die, the 6x6 lines give yields for the center 6x6 dies, excluding 16 dies around the edges, and the 4x4 numbers give yields for the center dies. These numbers show handling effects, showing that a spatial distribution with high defects around the edges is more accurate, except for Fab B which has higher defects in the center. The defect rates in Figure 4.4 are the 2x D_0 values for 6x6 dies. The D_0 and Y_0 values are least squares regressions of the 1x, 2x and 4x points using the Poisson yield formula.

VLASIC yields should be multiplied by a Y_0 factor. However, such a factor would either overpredict 4x yield or underpredict 1x yield, as the measured yields fall faster

than VLASIC yields for the larger structures. For example, multiplying the VLASIC Fab A 1x result (97.8%) by the measured wafer Y_0 of 0.94 produces a yield of 91.9%, which is not far above the measured 89% result. Multiplying the VLASIC Fab A 4x results (91.6%) by the same Y_0 produces 86.1%, which is significantly higher than the measured 77% result.

VLASIC overpredicts yield for the larger structures, even when multiplied by the Y_0 factor. Overprediction for the 4x structure ranges from factors of 1.09–1.12 for Fab A, 1.24–1.38 for Fab B, and 1.22–1.24 for Fab C. This differs from overpredictions for the 1x structures of 1.02–1.03 for Fab A, 1.06–1.1 for Fab B, and 1.07–1.09 for Fab C. This cannot be explained by modifying the systematic defect factor Y_0, or by clustering, which would raise rather than lower the simulated yield results. It appears that further work on yield models is needed if they are to accurately predict circuit yield.

Chapter 5

Soft Configurable WSI Yield

Circuit unit yields evaluated in the previous chapter are combined using various models to predict overall wafer yield for soft configurable designs similar to the pipelined memory prototype. Models are presented separately for site and switch yields, and then combined using Monte Carlo simulation. The chapter concludes with a yield model satisfying runtime reliability requirements.

5.1 Wafer Scale Yield

Wafer scale yield differs from die yield in the definition of what is accepted as a usable part. The traditional definition of yield is

the fraction of dies *that function correctly.*

Each die must be 100% perfect; a single fault renders it unusable. Fault-tolerant WSI designs are more flexible; they are designed to function correctly even in the presence of faults. So fault-tolerant WSI yield is

the fraction of wafers *that function correctly*

with or without faults. The flexibility introduced by fault tolerance is also present in this definition. Different requirements for "correct functionality" result in different wafer yields. For example, extra hops to bypass faulty switches may be limited by maximum latency requirements. Applications designed for a fixed memory size will not be able to use wafers with fewer than some minimum number of working sites.

Flexibility in yield definition is intrinsic to the use of fault tolerance to improve yield. One consequence is that it is not possible to use a single figure to describe yield, even when switch and site yields are available. Yield can only be measured with respect to some set of functionality requirements. We will concentrate on requirements for latency and silicon utilization.

Silicon utilization has several possible meanings. One is the amount of silicon devoted to application sites as a fraction of the total silicon area; this measures the overhead of reconfiguration circuitry. Another view of silicon utilization is related to the number of working sites. It could either be the number of working sites on a particular wafer, or the number of working sites required by the application. The definition used here is that

utilization is the fraction of sites required by the application.

This is a minimal measure that ignores any additional working sites, whether or not the application can use them. If an application can use extra sites, then other measures such as the expected utilization and its variance may be more meaningful.

Silicon utilization can be traded against yield. To see this, suppose that wafer fabrication produces 60% working sites on 25% of the wafers, and 40% working sites on 65% of the wafers. Then an application can improve its yield from 25% to 65% by lowering its utilization requirement from 60% to 40%. This tradeoff exists whenever an external application is designed with fixed assumptions about the wafer; if it can instead use any arbitrary number of working sites wherever they appear on the wafer then the tradeoff disappears.

Examples of fixed configuration requirements and their effects on yield are presented in the following sections. Yield curves are presented as if the configuration requirements

are truly fixed and cannot be changed. However, the flexibility in defining yield can be used to advantage by selling those parts with fewer working sites at a cheaper price.

5.2 Requirements for a working system

The simplest definition of a "working system" is one that has at least some "sufficient" number of working sites per wafer. This places no constraints on where these sites are located, so that the load (number of sites accessible) on each configured path may vary. Alternatively, one could configure paths with extra jogs to balance the load. This would cost in performance, but may save in ease of design at the next higher level.

This simple definition is used to produce the *baseline* curves shown in Figure 5.1 for a system with 8x8 sites. It shows the maximum attainable wafer yields for varying utilizations; all subsequent requirements are more restrictive. Baseline yield is calculated as the probability that at least some fraction f of the sites are functioning:

$$\text{baseline yield} = \sum_{i=N}^{T} \binom{T}{i} p^i (1-p)^{T-i}$$

where T is the number of sites per wafer (64 in Figure 5.1), N = $\lfloor fT \rfloor$ is the fraction required to work, and p is the probability that an individual site works.

Notice that in each case, yield is virtually 100% for high site yields, and then drops sharply for site yields below a certain point (dependent on the utilization). Every curve falls from over 95% to below 50% wafer yield within a range of site yields around 10%, with the higher utilization curves dropping even faster. For instance, the 70% utilization curve (44 of 64 sites required) falls from above 95% to below 50% wafer yield as site yield decreases from 77% to 67%.

Here and later, the 95% wafer yield point is used as an approximate reference for the beginning of a rapid fall in yield. This 95% figure has no particular significance other than being sufficiently below 100% to show a definite downturn in yield.

Systems are rarely so flexible that widely varying loads per path can be easily tolerated. We consider two additional constraints:

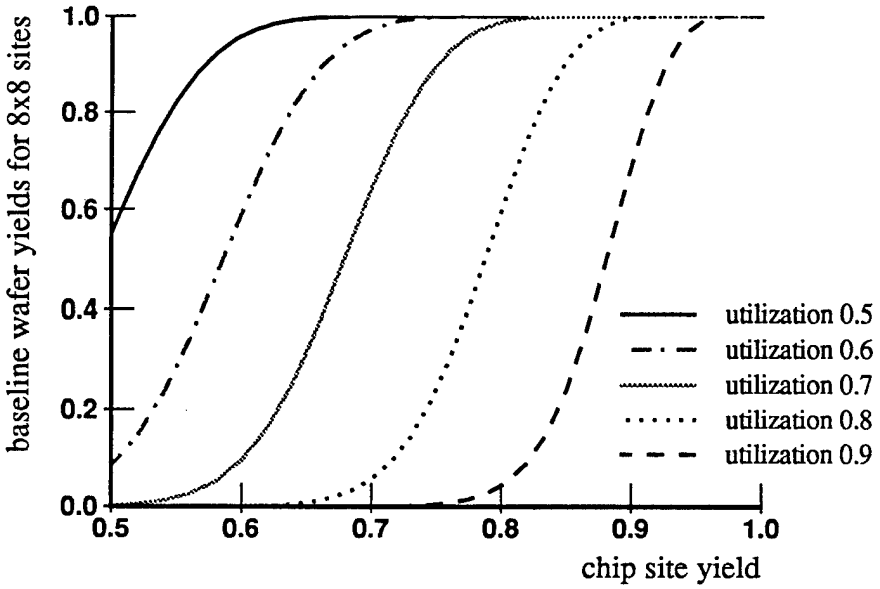


Figure 5.1: Baseline yields for different utilizations.

balanced loads: requires a minimum number of functional sites per path.

heterogeneous pairs: requires one of each pair of sites, assuming a different site type on every row, and that each path must have one of every site type.

Balanced loads eliminates wafers that, even with a sufficient total number of working sites, have a wide spatial variation in site yield. *Heterogeneous pairs* is the most stringent requirement, needing both balanced loads and 50% utilization.

The remainder of this section describes the resulting yields from these system requirements. Yield curves for balanced loads and for heterogeneous pairs are first presented separately (Figures 5.2 and 5.3). Following this (Figures 5.4 and 5.5) is a comparison against the baseline curve, for 8x8 and 16x16 fabricated sites at 50% silicon utilization.

The following assumptions were made to simplify the analysis:

- Switch paths are configured to access the working sites within pairs of adjacent

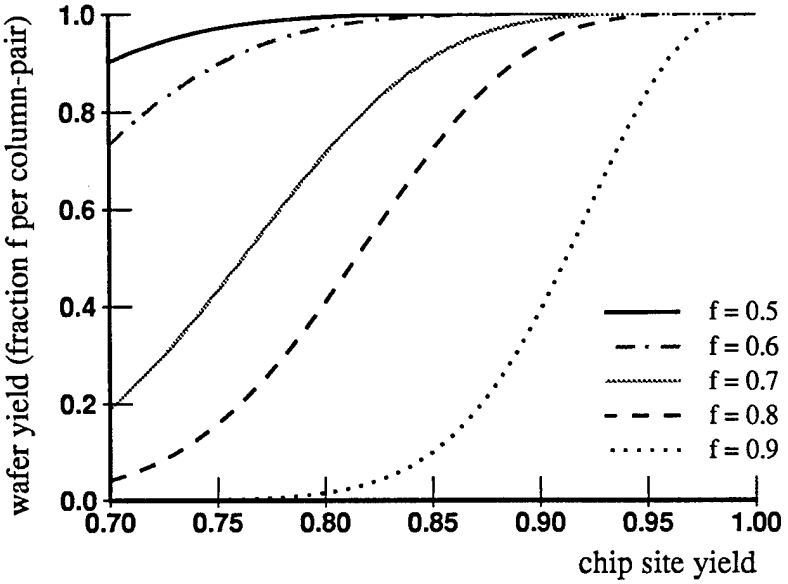


Figure 5.2: The *balanced loads* yield measure

site columns. Thus there is one switch path for each pair of site columns, and it does not access any sites outside these two columns.

- Each pair of site columns is surrounded by 3 columns of switches, of which enough are working so that every good site can be accessed. Recall that a site can be accessed by switches on either side. This assumption rests on the high switch yield (99.2%) predicted in the previous chapter.

- Faults do not cluster at the block level. The smallest block is a switch, which in this implementation contains 1266 transistors spread over around $2mm^2$. These figures include both the switch and the switch-to-switch interfaces.

Some of the curves examine the effect of relaxing system requirements by allowing one column-pair to be completely disregarded. As expected, this leads to higher system yield, at the expense of more wasted silicon (lower utilization) per wafer.

The fault-tolerant wafer yield of a system with balanced loads is shown in Figure 5.2

for a wafer with 8x8 sites. A wafer is functional according to the balanced load requirement when every column-pair has at least $m = \lfloor 2rf \rfloor$ working sites, where 2r is the number of sites in a column-pair, and f is the chosen fraction. Given a site yield of p in a system with r rows and c columns, wafer yield probability can be calculated as follows. The probability that at least m sites work in every column-pair is

$$\text{balanced load yield} = \left[\sum_{i=m}^{2r} \binom{2r}{i} p^i (1-p)^{2r-i} \right]^{c/2}$$

For instance, with 80% utilization (f = 0.8), a site yield of 93% (the 4K RAM) corresponds to a wafer yield of 98.5%, while a site yield of 79% (the 16K RAM) corresponds to a wafer yield of 34.7%.

The balanced load curves are similar in shape to the baseline, although wafer yield begins its rapid descent at higher site yield values. For instance, the 70% utilization curve drops from over 95% wafer yield to below 50% as site yields fall from 87% to 76%, at around 10% higher site yields than the equivalent drop in the baseline curve. The curve shapes are similar because in both cases, a constant fraction of sites are required, the only difference being that the former is per wafer, and the latter is per column-pair within the wafer.

In a system with heterogeneous pairs, an even more stringent requirement must be satisfied: that one of every pair is functional. Figure 5.3 shows the probability that this requirement can be satisfied for systems containing 8x8, 8x4 and 4x4 sites. Given the probability p of a working site in a system with r rows and c columns, the probability that at least one of each pair works is

$$\text{heterogeneous pair yield} = \left[p^2 + 2p(1-p) \right]^{rc/2}$$

These curves fall much faster than the balanced load curves. Note that even for as few as 4x4 sites, there is only a very short plateau before the wafer yield begins a rapid descent. Comparing this to balanced loads for the same silicon utilization (f = 0.5) and 8x8 sites, we see that a site yield of 0.7 results in a heterogeneous pair yield of 0.05 compared to a balanced load yield of 0.9. Unless very high site yields are probable, heterogeneous pairs will be an extremely difficult requirement to satisfy.

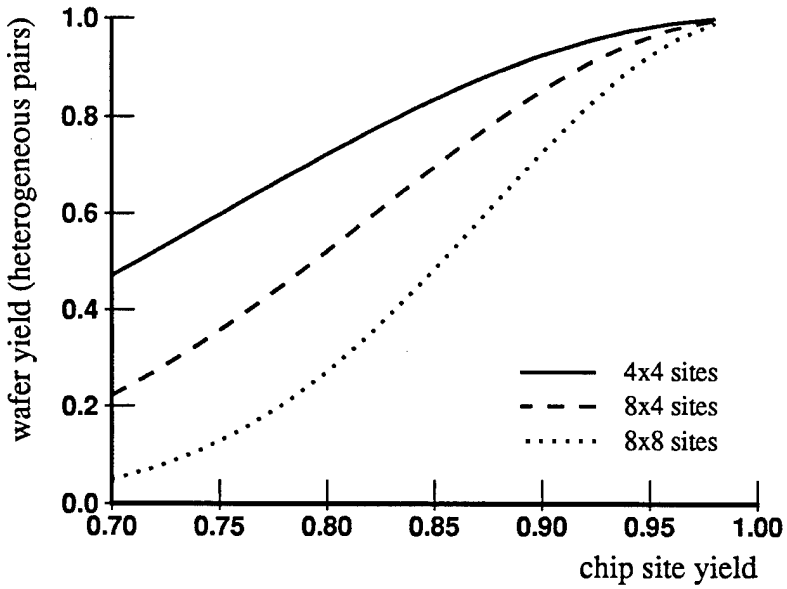


Figure 5.3: The *heterogeneous pairs* yield measure

Figure 5.4 compares different yield measures with the same utilization. The *baseline* curve assumes that all working sites can be used, irrespective of their location; providing the best possible wafer yield for a given site yield.

The *balanced loads* curve is taken from Figure 5.2. The *balanced except one column-pair* curve attempts to improve the latter by permitting one column-pair to be discarded, and increasing the fraction required for the other column-pairs to bring the overall wafer utilization back up to 0.5. However the higher utilization demand on the other column-pairs cost more than was saved by discarding one.

The *heterogeneous except one column-pair* curve requires one of each pair of sites in every column-pair but one. Note that utilization drops here, and cannot be scaled to 0.5 by any straightforward compensation. The final curve is *heterogeneous pairs* taken from Figure 5.3.

The *balanced except one column-pair* and *heterogeneous except one column-pair* curves are produced by applying the binomial formula twice. First, the probability P

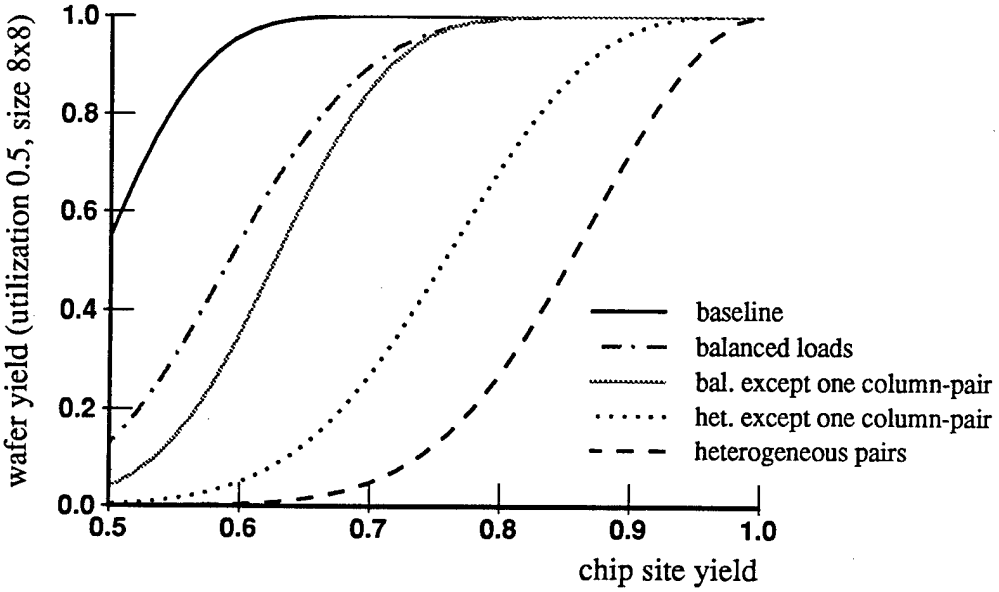


Figure 5.4: Comparing different requirements with the same utilization for 8x8 sites.

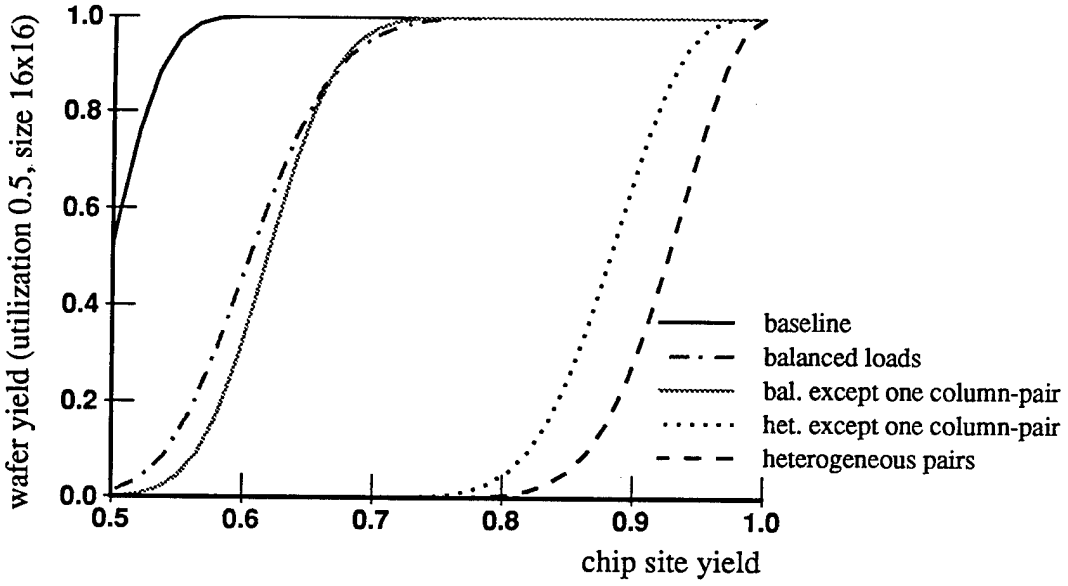


Figure 5.5: Comparing different requirements with the same utilization for 16x16 sites.

that a single column-pair works is calculated as before. Then, instead of raising this to the power c/2 (requiring every column-pair to work), the column-pair probability is substituted into another binomial formula

$$\text{yield less one column}-\text{pair} \;=\; d\,P^{d-1}\,(1-P) \;+\; P^{d}$$

to calculate the probability that all but one column-pair works or that all column-pairs work. Here d = c/2 which is the number of column-pairs.

Figure 5.5 shows the same yield models applied for 16x16 sites rather than 8x8. Note that the *balanced except one column-pair* now crosses *balanced loads*, however there is still no substantial difference between where the steep falls begin. Furthermore, wafer yields drop more rapidly than in Figure 5.4, and the *heterogeneous except one column-pair* curve falls even sooner, dropping below 95% wafer yield at a site yield around 95% rather than 89%.

It is apparent from Figure 5.4 and further accentuated in Figure 5.5, that the more global a requirement is, the better it can tolerate faulty sites. Very good site yields result in excellent wafer yield, whatever the model; a few bad sites are easily bypassed. However the crucial point at which site yield begins its descent shifts substantially as the models become less restrictive. In Figure 5.5, wafer yield drops below 95% when site yield falls below 98% for heterogeneous pairs, the most local requirement. For balanced loads, wafer yield remains above 95% for site yields as low as 77%, improving as the scope of the restriction lifts from 2 sites to 32 sites (contents of one column-pair). Baseline yield is a fully global requirement, maintaining wafer yields over 95% even longer, for site yields as low as 55%.

5.3 Switch Yield Limitations

What if switch yield is not really the 99.2% predicted in chapter 4? How is wafer yield affected by faulty switches? The above curves assume that there are an even number of columns c, divided into pairs such that each switch path can access 2r sites, where 2r is

the number of sites per column-pair. This requires enough working switches to configure a path past all working sites in each column-pair.

Wafer switch yield is the expected proportion of wafers with enough good switches. These wafers may or may not have enough good sites; we will combine site and switch yield prediction in the following section.

Performance goals may preclude the use of multiple bypasses around faulty switches. The extreme case of straight switch paths in every second column produces pessimistic wafer switch yields. For 8x8 sites, there are 4 switch paths, each containing 9 switches. With 99.2% switch yield (Psw), all 36 switches will work with a probability of 75%. A larger system with 16x16 sites will have a wafer switch yield of only 33.5%. The straight switch path yield is calculated as $Psw^{(r+1)*c/2}$. This formula assumes that input and output ports are fixed.

Higher wafer switch yields can be obtained with more flexible requirements. The requirement for straight paths could be relaxed to accept longer switch path latencies, which may vary for different paths within the wafer. However, latency could be a critical performance parameter, and designing an external system to accommodate varying latencies may be nontrivial. The requirement for fixed input and output ports can also be relaxed.

Figures 5.6 and 5.7 illustrate various ways of relaxing the straight path requirement that incur little or no extra latency. Figure 5.6 shows how switch column-pair yield varies as a function of switch yield, either with straight paths or with a single bypass. The bypass adds one cycle latency where it is required, for extra flexibility within the switch path.

Figure 5.7 uses the column-pair yield of Figure 5.6 together with the number of columns to predict wafer switch yield. Extra columns of switches and sites add flexibility to the choice of input and output ports to avoid paths containing faulty switches. These curves predict wafer switch yield given a switch yield, path latency, and the number of redundant columns. The following paragraphs describe them in more detail.

Switch column yield (Figure 5.6) falls fast for straight paths, so is very sensitive to

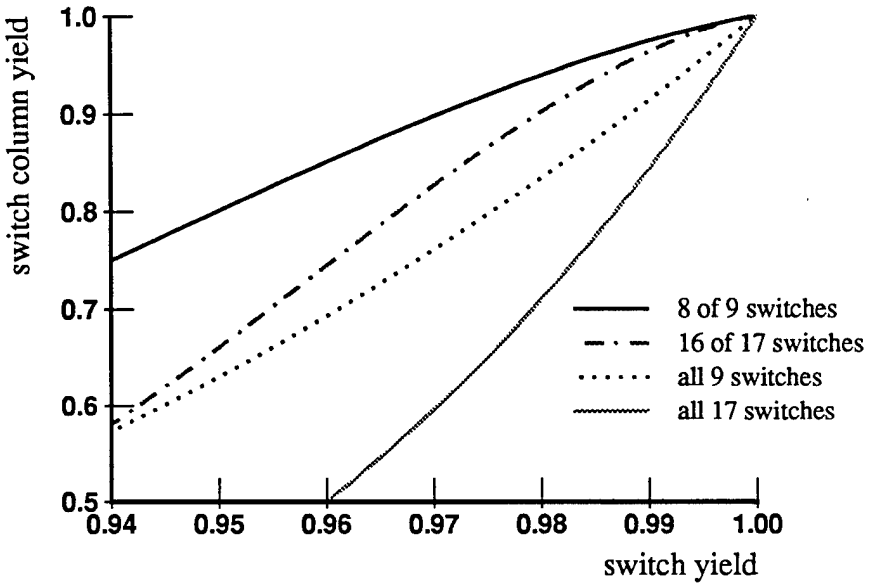


Figure 5.6: Switch column yield as a function of switch yield.

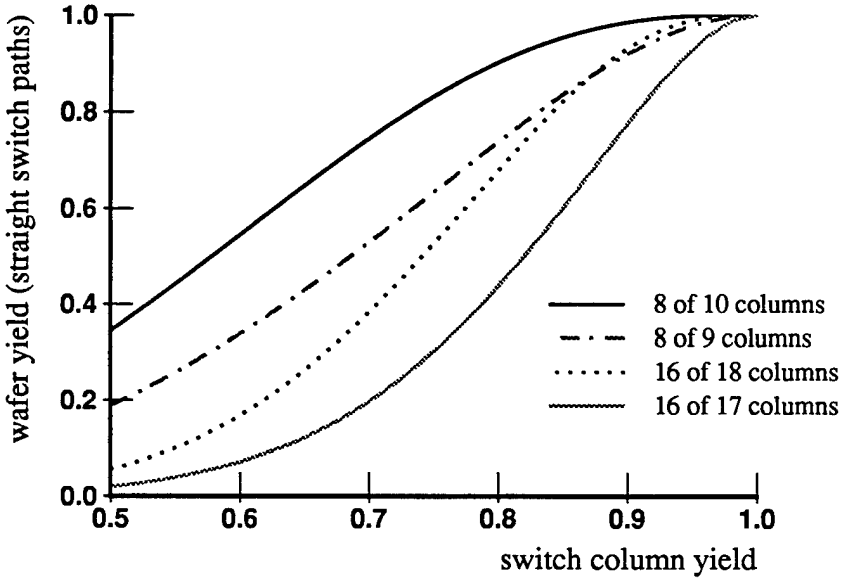


Figure 5.7: Wafer switch yield as a function of switch column yield.

small changes in switch yield. For a simple switch with a high yield of 99.2%, requiring straight switch paths with 16x16 sites (and 17 switches per path) results in a switch column yield of 87.2%. Figure 5.7 shows this to result in a wafer switch yield of 96.9% for 8 of 10 site columns, or 87.4% for 16 of 18 site columns. These are a considerable improvement over the earlier results of 75% and 33.5% for straight paths with no spares and no port choices. Thus even with straight switch paths, wafer switch yield can be brought to an acceptable level by providing redundant switch and site columns.

Figure 5.6 shows switch column yields both for straight switch paths and for paths with a single jog. The upper two curves fall more gradually because of the flexibility introduced by permitting jogs. It is assumed that only internal switches are bypassed, so input and output switches must always work. If a switch fails, then all 3 switches on both sides are required to work so that both nearby sites can be accessed by the bypass (this is really a fork and join in the switch path). The bypass curves are approximations; they do not allow for bypasses of multiple faulty switches (if done with a single bypass, latency remains the same) or for conflicts where one or more switches are needed for bypasses by both neighboring switch paths.

Extra site columns add flexibility in positioning the column-pairs. Figure 5.7 shows the resulting improvements in wafer switch yield. Column-pairs are accepted whenever they contain enough good switches to satisfy the latency requirement (zero or one jogs) and to access all working sites. Edge paths are ignored, since they can access only one column of sites.

The curves in Figure 5.7 drop less steeply than those of Figure 5.6. Nevertheless, they provide very little design leeway; high switch column yields are necessary for good results. Even in the best case shown (8 of 10 site columns required), a switch column yield of 80% is required before wafer switch yield exceeds 90%. We show in the following section that allowing switch bypasses can greatly improve the switch column yield and the resulting wafer yield.

The curves in Figure 5.7 are produced using the formula

$$\text{wafer switch yield} = \sum_{j=0}^{\lceil k/2 \rceil} N_k(j)(1 - Pc)^j Pc^{k-j}$$

where Pc is the probability that a switch column-pair is functional, k = c–1 is the number of internal switch columns (ignoring the two edge columns), j is the number of faulty switch columns, and $N_k(j)$ is the number of arrangements of those j faulty columns from which the appropriate number of site column-pairs can be accessed. This last function is easily calculated for the relatively small numbers under consideration by generating and checking every possible arrangement of the j faulty columns. The number of site column-pairs needed is (c-1)/2 for one extra site column, and (c-2)/2 for two extra site columns.

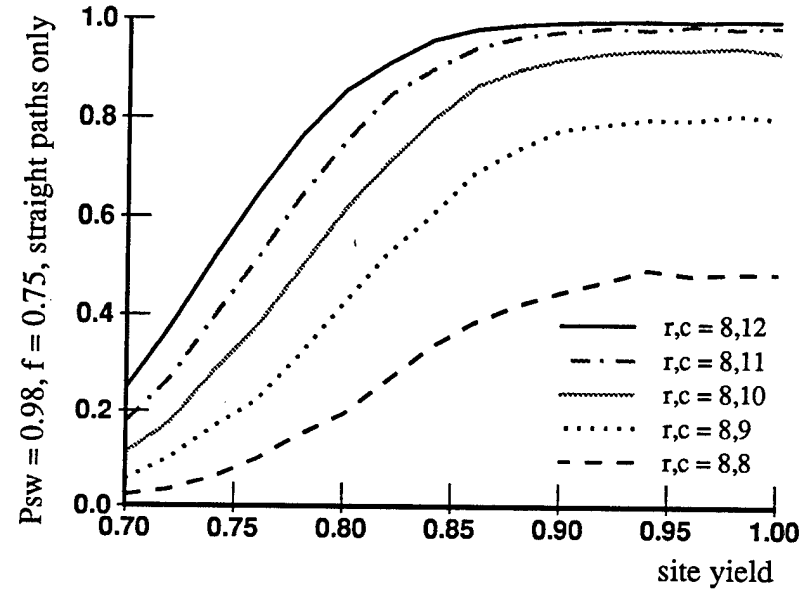
5.4 Combining Switch and Site Yields

Spare site and switch columns can be shared, replacing parts of the wafer with faulty sites or switches as needed. This sharing removes independence assumptions required to use the simple yield formulas above, so wafer yields for the balanced load model were produced using Monte Carlo simulation. Sites and switches are randomly assigned to be faulty according to their respective yield values, and on each iteration the simulator tries to find enough switch paths with access to the minimum required number of working sites.

I simulated at least 5000 iterations per data point, followed by groups of 500 iterations until the yield changed by less than 0.0008 (absolute). The results show the effects of varying the number of spare columns and the switch yield. The first group (Figures 5.8 and 5.9) use straight switch paths only. The second group (Figures 5.10 and 5.11) allow arbitrary bypasses within the 3 switch columns surrounding each pair of site columns.

Figure 5.8 shows wafer yields when the number of spare columns varies, and switch paths are straight. Clearly, 98% switch yield severely limits yield for the larger wafers with 16 rows. There is a significant improvement with each additional spare column. With 8 rows, additional spare columns beyond 2 add little improvement.

Figure 5.9 shows wafer yields when the switch yield varies, and switch paths are straight. These curves show how sensitive wafer yield is to small changes in switch



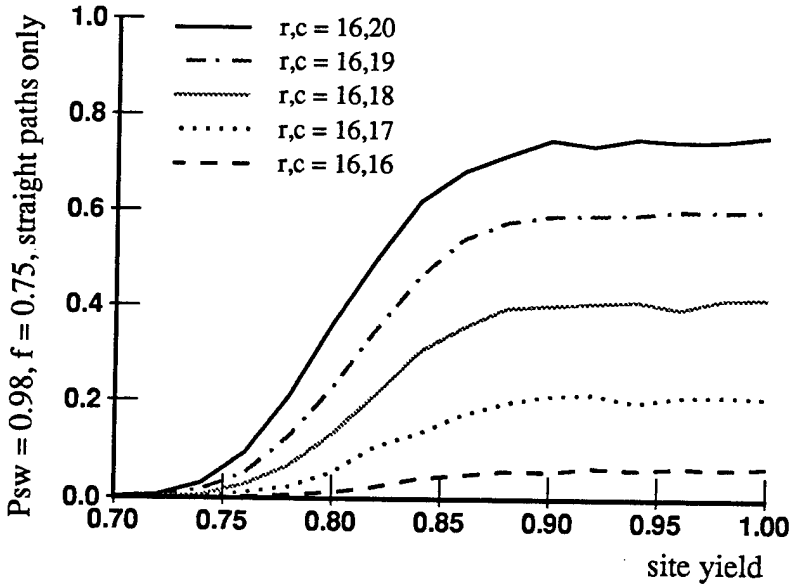


Figure 5.8: Varying the number of spare columns, with straight switch paths. Psw is the switch probability, f is the fraction of sites required per column-pair, r and c are the number of rows and columns, and (c - r) is the number of spare columns.

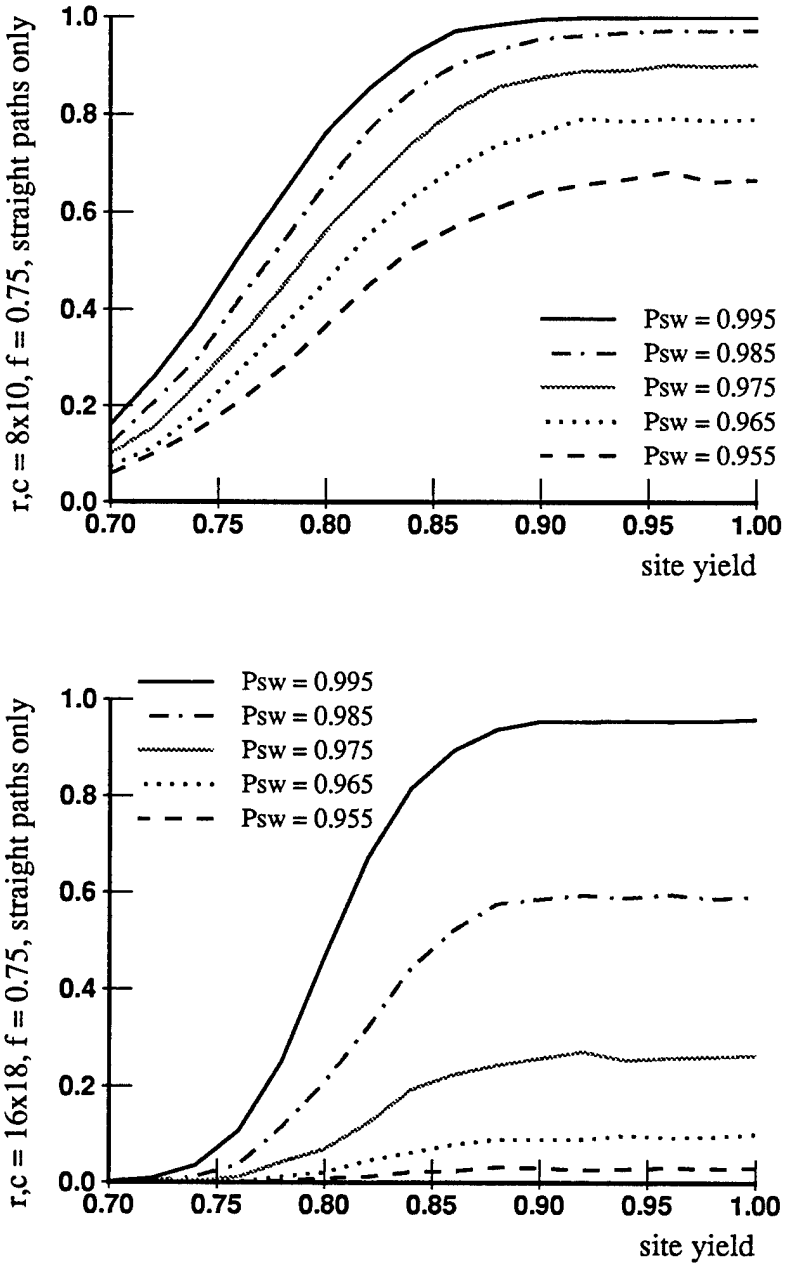


Figure 5.9: Varying the switch yield, with straight switch paths. Psw is the switch probability, f is the fraction of sites required per column-pair, r and c are the number of rows and columns, and (c - r) is the number of spare columns.

yield. This sensitivity is much more extreme for the wafers with 16 rows. Switch yields must exceed 99% for wafer yield to reach acceptable values, without providing more than 2 spare columns.

Figure 5.10 shows wafer yields when the number of spare columns varies, allowing switch path bypasses. These curves are not limited by the 98% switch yield; 2 spare columns provide wafer yields over 99.7% once site yield exceeds 90%.

Figure 5.11 shows wafer yields when the switch yield varies, allowing switch path bypasses. Notice that switch yields vary down to 89.5%, rather than 95.5% as in Figure 5.9. With bypasses, switch yield is much less critical, providing wafer yields over 98% for switch yields over 95.5% and site yields over 90%. This provides a reasonable margin for switch design complexity.

Notice that in every case, wafer yield rises to a plateau. The plateau level is limited if there are too few spare columns, or if switch yield is low. The larger wafer with 16 rows has a longer plateau followed by a steeper drop, as shown in earlier sections. Compared to the site data without switch yields, these curves have a lower maximum, dependent on the switch yield, and fall faster as switch yield loss eliminates access to functioning sites.

Switch path bypasses add one or more extra clock phases of delay. Odd numbers result from bypasses at the input or output port (but not both). Initial estimates of the number of switch paths requiring multiple bypasses indicate that the vast majority need no more than two or sometimes three extra phases. Shared switches, when neighboring switch paths have bypasses using the same switches, were not checked. External software can ensure that data is not sent at the same time to two switch paths when a shared piece has a conflict.

In the previous section, bypasses were assumed to be two sided, and to bypass only one switch within the path (not a port). These pessimistic assumptions produced low wafer switch yields. The Monte Carlo simulations check for every possible bypass, including bypasses of multiple faulty switches, and half bypasses where only one side has enough good switches, but there are nevertheless enough good sites accessible using

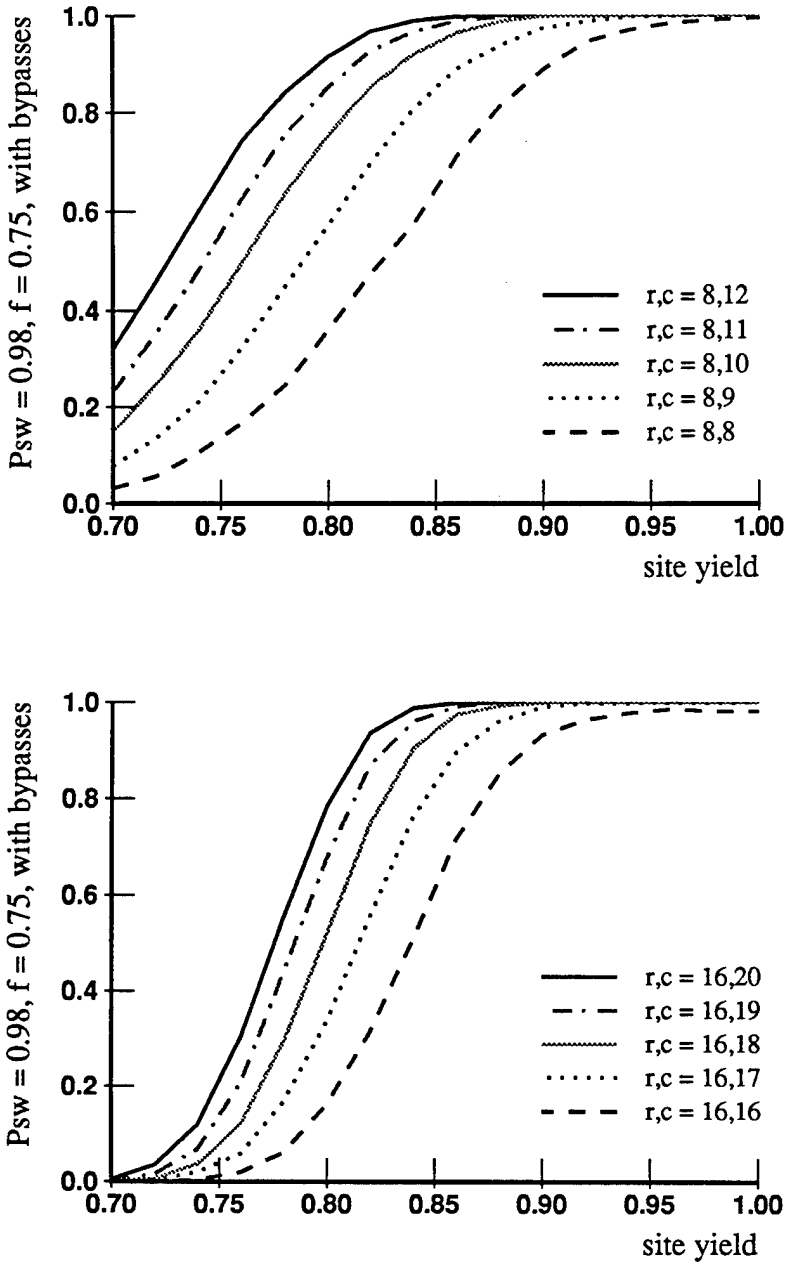


Figure 5.10: Varying the number of spare columns, with switch path bypasses. Psw is the switch probability, f is the fraction of sites required per column-pair, r and c are the number of rows and columns, and (c - r) is the number of spare columns.

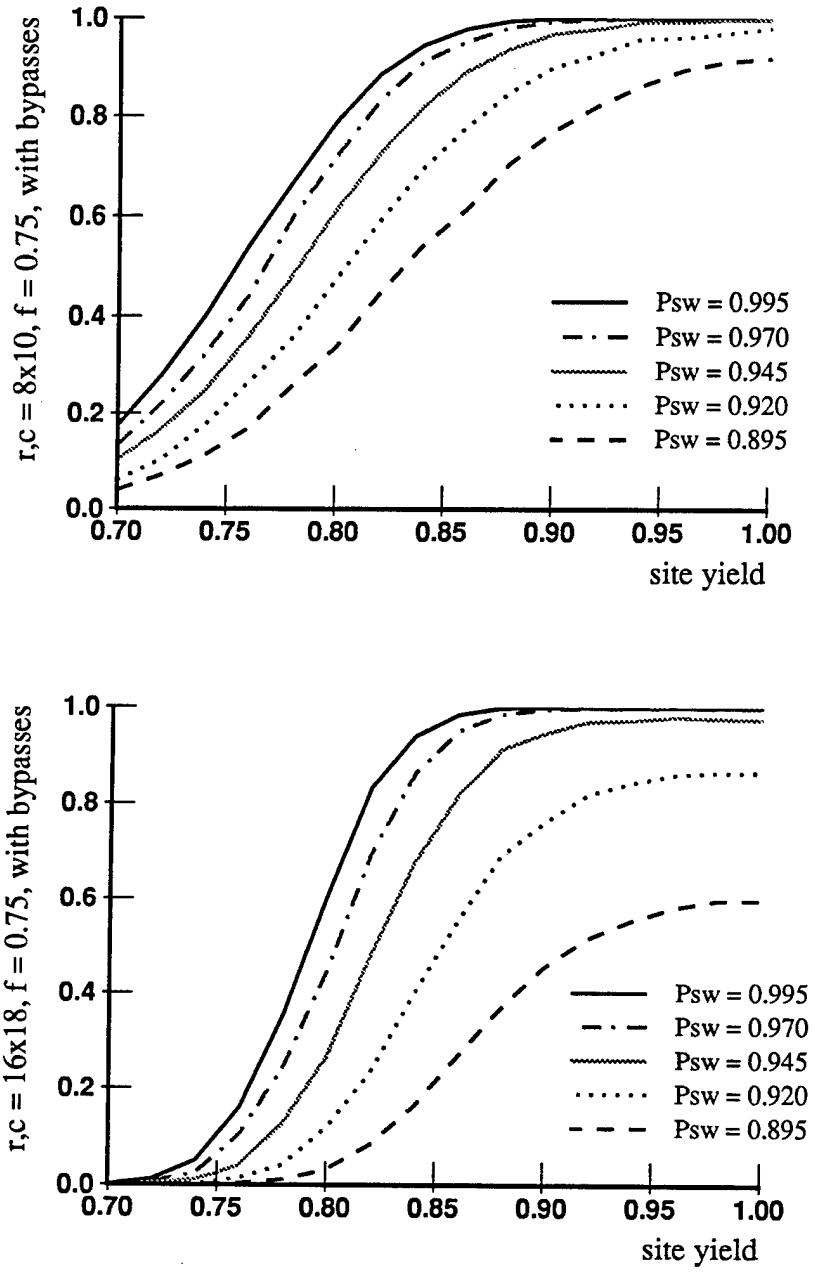


Figure 5.11: Varying the switch yield, with switch path bypasses. Psw is the switch probability, f is the fraction of sites required per column-pair, r and c are the number of rows and columns, and (c - r) is the number of spare columns.

only one side. Port bypasses are permitted, but required to be half bypasses, even if both nearby switches are working. This is based on the assumption that the external interface supports a predetermined number of switch paths per wafer.

Bypass optimization selects the switch path that minimizes additional delay. Coalescing multiple bypasses reduces the number of wriggles. Single as well as double bypasses can be coalesced in those cases where the join is short enough so that there are enough good sites accessible. When both switches near a faulty port are functioning, the half bypass can be chosen to minimize delay. Optimized paths combined with Monte Carlo simulation will show wafer yield as a function of maximum bypass delay, where the maximum allowable delay is set to 1, 2 or 3 clock phases.

Various examples of bypasses are shown in Figure 5.12. Notice that when half bypasses are joined with full bypasses, only one side has a reduced delay, and external software may be complicated by the difference in delay between switch data arriving at each side of the join at the top of the full bypass.

5.5 Runtime Fault Tolerance

Runtime fault tolerance for sites can be handled with any selection of paths and loads using a simple scheme. Divide the number of sites accessible from the path into pairs, discarding the odd one if necessary. Assign the same (soft-settable) address to both sites in each pair. Disable the outputs of one in each pair by resetting the mask bit of the switch that could read it. Then both sites will receive every command, hence contain identical information. Should testing of this site address reveal an error, reconfiguration would disable the current site output mask bit, and enable the silent backup site.

This works independently of the location along the path of the two sites, provided that site outputs are timed so as not to squash commands that have reached one but not the other of a redundant site-pair. If collisions are likely to be a major concern, then a routing scheme that allows congestion may be preferred. Collisions can always be avoided by waiting an extra cycle or two before sending the next command.

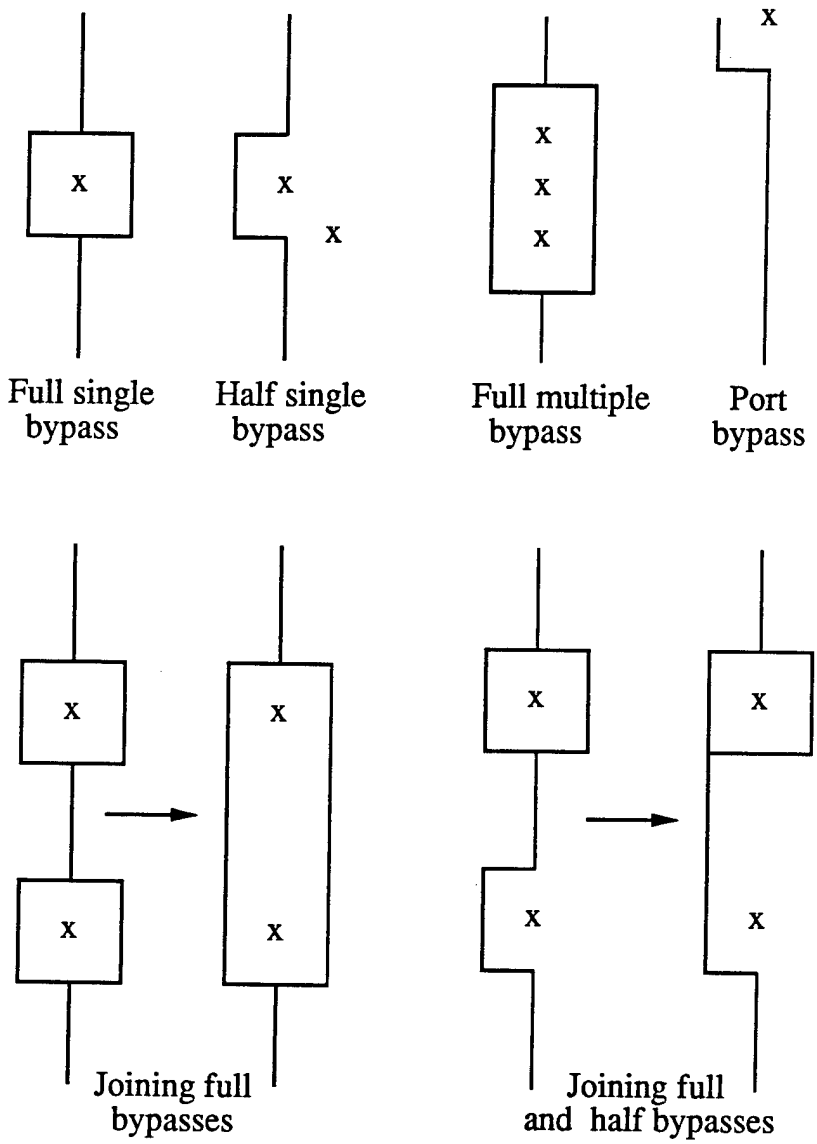


Figure 5.12: Switch path bypass examples. Faulty switches forcing bypasses are shown with an 'x'.

To evaluate the yield of this scheme, the reader is referred to Figure 5.2 which shows curves for balanced loads per path for 8x8 sites. The actual number of usable sites would be half of those available — to allow for the silent backup. For instance, if 5 sites per column-pair are required, together with 5 backup sites, the utilization needed is 10 out of 16 sites which is closest to the 60% utilization curve. For this curve, a site yield of 0.75 corresponds to a wafer yield of 0.9.

Periodic runtime switch tests will ensure that the switch path remains functional. Runtime faults are then bypassed by rerouting to use other switches. While periodic testing exhaustively verifies every switch connection, another much simpler test is to monitor the output switches. The output switch should track the input exactly except in cycles containing read data generated by the sites. Such monitoring has the advantage of not interrupting normal system operations.

5.6 Summary of Soft-Configurable Yield Models

The flexibility introduced by fault tolerance is reflected in the models used to evaluate wafer scale yield. Wafer yields vary considerably for the identical wafer evaluated using different models.

Heterogeneous pairs is shown to be too restrictive to be of much practical use, even when relaxed by permitting a redundant column-pair. Balanced loads are considerably better, showing that moderate distribution requirements can be readily satisfied. The best wafer yields are obtained through the baseline model, which makes no distribution requirements. In all cases, wafer yield was excellent (very close to 100%) for good site yields, before dropping sharply at a certain critical point. These plateaus of high wafer yield provide a good design margin so long as site yield can be tuned well past the critical point.

Switch yield is more critical, requiring spare columns or switch path bypasses for design margin. Switch circuitry and layout should be tuned carefully to maximize yield. Such tuning is easier for simple switch designs.

Wafer switch yield is heavily dependent on the number of rows (number of switches in a switch path). Wafer switch yield can be improved by squashing sites into fewer rows, each of which are longer. The tradeoffs here include extra I/O ports, and longer power buses (or some alternative to feeding power by regular metal along the rows).

Monte Carlo simulations show that dependence on high switch yields can be reduced substantially using switch path bypasses and/or spare columns. The simulated curves show the best yield available using switch path bypasses where needed to cover the required number of working sites within each column-pair. Switch and site yields combine to produce wafer yield for the balanced load model, using spare columns to replace faulty sites or switches as needed.

The preceding section describes a model for the yield of a system with runtime redundancy. This is a simple adaption of the balanced yield model. Silicon utilization is traded by using two working sites to provide the same data for each site address.

One issue that has not been considered in this chapter is how to choose the best site size. An approximate model for this is presented in Chapter 7.

Chapter 6

Related Work

We survey several recent WSI projects. These include alternative approaches to soft configuration, some hard configured systems, and voting represented by the Trilogy project.

Hybrid multichip carriers provide high circuit density without either fault tolerance or device scaling. We review two multichip modules used in commercially available products.

6.1 Soft Configured Systems

The three projects described here have all produced working prototypes. Anamartic is the only one to progress beyond this stage to commercial production. Their 200 Mbit memory is particularly notable because, of all the WSI projects over the last couple of decades, it is the first and only one to reach the marketplace.

The WASP and ELSA designs both use soft-programmable switches to configure processor arrays for systolic applications such as image processing.

6.1.1 Anamartic 200 Mbit dRAM

The only wafer-scale product currently on the market is produced by the British company Anamartic in collaboration with Fujitsu[5]. A wafer of 1 Mbit dynamic RAM sites is configured into a serially accessed spiral. The average access time is 200μs, and the average transfer rate is 1.6 Mbytes/s.

The soft-configurable controller, referred to as the CONLOG, contains 1200 gates, adding 20% overhead to the area of each 1 Mbit dRAM site. Total site area, including this overhead, is 13.65 x 4.4 mm^2. Power, clocking and a command strobe are distributed globally to all CONLOGs. Each CONLOG has 4 1-bit I/O paths to its 4 adjacent neighbors. All data and controls are transferred serially. The CONLOGs are used to configure a spiral path through the working parts of the wafer, so they can all be accessed through a single external port. The global signals are driven separately from the top and bottom of each column, with a split in the middle.

The wafer carrier board holds a non-volatile flash PROM containing the locations of working dRAM blocks, indicating which are spares and which are in use. A 1 Mbit site contains 32 blocks of 32 Kbits, each of which are separately enabled. Only working blocks are accessed. The memory is organized as 256K x 4 bits. The CONLOG uses a power switching transistor to disable dRAM power in the event of a power short in any 32 Kbit dRAM block.

A 6 inch wafer contains 202 sites, for a maximum of 202 working Mbits. Two wafer-carriers are hermetically sealed face-to-face to form a 40 Mbyte module, requiring 80% of the available memory blocks. Each controller can use 32 Kbit blocks from one wafer to replace faults in the other, providing the full 40 Mbytes are available. Four modules can be stacked to provide up to 160 Mbytes. The system operates at 20MHz, using 42W when active, and 10W on standby.

The goal is to provide a cost-effective step in the memory heirarchy between main memory and disk. The initial cost for limited quantities is $11,680 for 40 Mbytes, and $28,760 for 160 Mbytes. The latter reduces to around $180 per Mbyte, still substantially

higher than the $3–$5 per Mbyte for conventional disks[21]. Their edge over conventional disks is the much faster access time combined with a slightly cheaper price than conventionally packaged boards of dRAM chips. With 90% fewer wire bonds and solder joints than the conventional equivalent, and no moving parts, wafer modules should be more reliable than either printed circuit boards or magnetic disks; while their currently specified MTBF (mean time between failures) is 50,000 hours, this is expected to rapidly increase to 200,000 hours or more. Thus companies (such as Tandem) who emphasize very high reliability are the most likely customers.

By using smaller granularity (32 Kbytes rather than 1 Mbytes), Anamartic produces higher yields than conventional single chip dRAMs at comparable points on the learning curve. When conventional yield reaches 20%, Anamartic claims yields of around 80% from the same technology, showing that high yield can be obtained on large (6-inch) wafers with sizable sites, provided that the sites have internal redundancy.

The spiral configuration limits performance, while allowing a low cost single port interface. A soft-configurable system of the type described in this thesis would require two ports to access all the sites: one input at the bottom, and one output at the top. The wafer could be configured with one input switch sending commands through parallel paths up every functioning switch column, joining at the top to reach the output port. The parallel path has delay proportional to r + c, where r is the number of rows and c is the number of columns, compared to delays averaging r * c to send and receive data through the Anamartic spiral. This performance improvement may justify the inclusion of one extra port.

6.1.2 WASP — A WAfer-scale Systolic Processor

The WASP systolic machine[19, 18, 20] is designed to hold over 400 16-bit processing elements (PEs) on a wafer, surrounded by a lattice of 8 switches per PE, or over 3200 switches. Switches and sites have connections to their 8 nearest neighbors (both orthogonal and diagonal). Connections are bidirectional, with 8 input and 8 output lines. A switch has degree 8 and width 16, with 16 bits of configuration memory, pass transistors

to steer the data, and a buffer to boost the signal. Circuit switched path delays are limited by partitioning the wafer into building blocks of 12 PEs, from which 4 working PEs can be configured with 99.5% probability, according to initial estimates.

A WASP prototype on a much smaller scale has been implemented[18]. It contains a 3 x 3 array of PEs surrounded by 40 switches, connected only to the N,S,E,W neighbors. The prototype PE is a bit serial processor that can produce 16 boolean functions from 2 inputs, and contains a single accumulator. The prototype switch passes one 4 bit input to a selection of the other three 4 bit ports (all bidirectional), through pass transistors controlled by an 8 bit register. The prototype was implemented in single metal 4μm nMOS through MOSIS. The PE size was limited by the need to fit within a readily available package. The prototype die was 6.3mm x 5.9mm, containing 9 PEs each with 203 transistors, and 40 switches each with 70 transistors.

The use of single metal technology led to significant difficulties in both global signal distribution and switch layout. These difficulties can be eased but not erased by the use of more recent technology with more metal layers. The global signals, switches and wiring network consumed around 50% of the die. This figure was estimated to drop to below 15% for a design with full 8 or 16 bit PEs. The PEs had more complex control requirements than were anticipated, despite deferring much of this to external signals. There were 10 fully global controls, and 3 global to each row of PEs, in addition to power and clock signals. Global lines were not fault-tolerant, though future design plans included reducing their number by half, and repairing through laser configuration.

6.1.3 ELSA — European Large SIMD Array

ELSA is a project of the European ESPRIT research cooperative. It is a 128 x 128 SIMD (single-instruction, multiple-data) array for image processing[42, 41, 40].

The ELSA processing element (PE) contains two 64-bit registers, a 1-bit shifter, a 1-bit ALU, and an inhibit flag. The shifter unit is used to load instructions along the PE columns, and and also to read and write the registers. These PEs are grouped into blocks of 12x7 to construct sites with one spare column. The grouping reduces global

configuration overhead by amortizing the cost over the 72 PEs per site.

Each site is surrounded by a double rail network containing 8 soft-configurable switch nodes, and 4 buffers. Connections between sites are circuit switched. Each switch has 6 transmission gates that are used in 4 modes:

Open: all paths disconnected

Rdiag: connections between N and W, and between S and E

Ldiag: connections between N and E, and between S and W

Cross: connections between N and S, and between E and W

The switch mode is stored in a 2 bit register.

Modes are configured from the periphery. On power-up, they are initialized Open (disconnected). The site PEs are set in "transparent mode" to shift the configuration through (vertically and horizontally) without performing any ALU operations. A programming signal is shifted in parallel to indicate when the data has reached its destination. The mode register is first loaded when exactly 1 of the 4 programming inputs is high, and that input is used as the source of all future programming.

After configuration, the mode register is fused to its final value. Clock, power and global signals are protected by fuses, and power layout is designed so that power and ground do not overlap.

The ELSA design provides little protection against switch failures, as a site must have 4 good switches surrounding it to be usable. Also, switch configuration requires the correct operation of the site PEs in transparent mode, to pass through the configuration requests. Although each individual switch is simpler than the soft configurable equivalent, the design requires 8 switches and 4 buffers per site. Laser configuration provides gross protection against switch and global signal failures, while eliminating the possibility of run-time reconfiguration. Lasers were chosen, however, to remove the need for configuration on every power-up.

6.2 Hard-wired configurable designs

Hard-wired systems are not directly comparable to soft designs, as they cannot be used for runtime reconfiguration and require more expensive manufacturing equipment (typically lasers). For systems that do not require runtime flexibility, hard-wiring has the potential for higher performance. Systems relying only on fuses have no special connections, and those with antifuses still provide lower resistance connections than transistors. The hard-wired examples illustrate the current state of research into this approach to wafer scale integration.

Once again, the first example is a commercial product. INOVA produces large static RAMs reconfigured to produce the next generation size in monolithic form. INOVA has produced wafer scale prototypes, but none of these have yet been marketed commercially.

The Lincoln Labs RVLSI (Restructurable VLSI) project has produced several working wafers for research and military use in small quantities, using lasers to form connections as well as to break them.

The Hughes project has working prototypes of an image processing system consisting of 5 wafers stacked vertically. They are currently working on a 32 wafer stack. Hard-wiring has been deemed insufficient for this design, leading to recent work on superimposing a form of soft configuration over the existing 32 wafer design.

6.2.1 Inova Static RAMs

Inova[46, 8] sells high performance static RAMs. In addition to column and row redundancy within each block, wafer scale techniques allow selection amongst multiple redundant memory blocks. This provides a lead time of at least a year over competitive single block implementations of the same size.

The 1 Mbit monolithic SRAM holds over 5 million transistors in an area of around 320mm^2. This product comes in configurations of 128K x 8 or 64K x 16, in 32 and 40 pin JEDEC standard outline packages. It is constructed using a double poly double metal 1.2μm process with a cell size of 108μ^2 and a die size of 9.1mm x 34.1mm.

Laser blown fuses in first level polysilicon are used within each block and to choose working blocks. Blocks are arranged in 2 rows of 20. Each block contains 32 Kbits with 2 spare rows and 2 spare columns. Pairs of blocks are tested independently after first metal. Faulty or extra blocks are disconnected by blowing fuses, before applying a final nonredundant layer of metal with 20μm pitch.

This part was on the market a full year before any competitors were announced. It runs at 55ns, 70ns or 100ns and remains the only monolithic 1 Mbit SRAM to conform to military standard 883C (in January 1990). Although not as fast as combining smaller size SRAMS of some competitors, it has sold to some 400 different customers and is currently selling in tens of thousands of units per month.

Future plans include both evolutionary improvements (4 Mbit parts with faster access times using 0.8μm technology) and full wafer scale versions in collaboration with Westinghouse. Future versions will delay fusing until after the second metal layer. Product yield to first metal was shown to correspond fairly closely with their model[8]. First metal yield has reached the planned 80% level, with the highest monthly average reaching 90%. However second metal yields have not yet reached the planned 95% level, though some lots have averaged 94%. There have been a number of 100% perfect wafers, where every die contains enough good blocks to configure 1 Mbit.

A full wafer 8 Mbit device was demonstrated in October 1989. This consists of 8 adjacent working 1 Mbit parts attached to a Westinghouse carbon fiber composite base with a high thermal coefficient, and wire bonded. Westinghouse plans to eliminate all solder and wire bond connections by using multilayer flat cable to connect to gold bumps on the wafer. A unit containing 2 such wafers, 2 flat cable layers, and a special elastomer in between will be connected through physical compression. External cable connections will be provided at one end of the package. The Westinghouse demonstration also included a Lincoln Labs RVLSI device (as described in the next section).

6.2.2 Lincoln Labs RVLSI

The Lincoln Laboratories Restructurable VLSI program has been running since 1979,

producing several different signal processing wafers together with a suite of computer aided design (CAD) tools to aid in their design[39, 6, 48, 11, 14, 3].

An RVLSI project is partitioned into a small number of site types. Many instances of each site type are placed on the wafer. Sites are designed to have around 50% yield. A grid of wafer length wiring tracks surround the sites, but are initially unconnected for site testing. After probe testing every site and interconnect wire, placement and routing tools direct the connections of sites and wiring tracks using laser links.

Wafer length interconnects are designed to be wider than the minimum to lower their resistance. They are tested for both shorts and opens by measuring their capacitance. Decoupling capacitors are provided within each wafer in restructurable units. Steppers are used to construct wafer size masks using at least 9 different sub-patterns. After configuration, wafers are placed in a metal box that can be hermetically sealed.

Laser linking is performed using two different technologies: the vertical laser link or the laser diffused link. The vertical laser link is formed by melting a layer of silicon nitride between two metal layers. Connections have around 10 ohms resistance for pads of length 22μm on a side.

The laser diffused link is produced by lateral diffusion joining two implant regions. This kind of link can be used with standard MOS processing, however has a larger resistance of around 100 ohms for a join area of length 16μm and width 4μm. Both forms of connections are extremely reliable over long periods of time, and very high yielding.

CAD tools were developed to route the working wiring tracks, to place the system into the working cells, to decide how many tracks are needed between cells, and to control the operation of the laser.

Six different signal processing wafer designs have been produced. Four of these use the same initial design, but apply laser configuration to specialize the function of the sites as well as to configure around faults. They contain linear or two-dimensional arrays of cells implementing functions such as Fast Fourier Transform, Hough Transform, 2-dimensional convolution, and speech recognition.

One recent design is the Lincoln Programmable Image Processing Wafer[6], which is a 4x4 systolic array of 16 bit processing elements (PEs). The wafer has 40 PEs and 40 memory cells in alternating rows of length 10. The PEs hold 28K transistors in an area of 5x4.5 mm^2 which includes an 8x8 array multiplier and a 128x16 bit register file. The memory cells have 2Kx8 bit static RAM requiring 140K transistors over an area of 5x8 mm^2 with 2 spare rows and 2 spare columns. Interconnect width is 54 horizontal tracks and 36 vertical ones. Decoupling capacitor cells of 1000 pF are included on the wafer. Each one is 0.5x2 mm^2 and fits entirely underneath the power lines. Only fault-free capacitors are connected to the system. This design has recently been received from fabrication, and at least one wafer contains enough good sites to configure a system.

6.2.3 Hughes 3-D Computer

The 3-D Computer project[28, 29, 24, 49, 10] aggressively reduces system size both by using wafer scale integration and by stacking wafers directly on top of one another. Cellular processor arrays are implemented using a novel approach. Processing elements are split into smaller functional units, with each wafer containing only one such functional unit. Intra-PE communication runs through the vertical bus. Thus PE complexity is limited by the stack height rather than the wafer size.

A 5–wafer stack has been implemented, and a 32–wafer stack is currently being fabricated. With 2 mil vertical spacing, a 1 inch stack could hold 45 wafers.

Vertical connections are implemented in two stages: connections between wafers and vertical paths feeding through each wafer. The connection between wafers is formed from microspring bridges. A bridge is a length of copper that attaches to the wafer at each end, and is disconnected in the middle by dissolving the spacer under the middle part. The bridge from the top wafer runs orthogonal to the bridge on the bottom to allow plenty of leeway for alignment. Bridges are needed for mechanical flexibility in the tall wafer stacks. Each one is 25 mils long and 3 mils wide. A path through 2 orthogonal bridges has a resistance of 2 ohms.

The feedthrough paths through each wafer are formed by applying a thermal gradient

to aluminum dots that migrate through the silicon. A conductive area of p-diffusion is implanted on the bottom side of the wafer to join the end of the migration path. These paths have a capacitance of around 0.2pF and a resistance of 2 ohms. They can be placed on a pitch of 10 mils. Feedthrough processing is carried out prior to fabrication; the microspring bridges are built by extra processing after fabrication.

The redundancy scheme for the 5 wafer stack consists of redundant cell pairs placed at each end of the microspring bridges. If either of these fails after fabrication, then it is disconnected using an ultrasonic cutter prior to the addition of the bridges. Driver cells are also duplicated, on each side of the wafer. Cell circuitry is placed directly under the bridges, so cell size is not unduly constrained by the 3-D connection procedure. Each wafer is a 1 inch square containing 32x32 PEs, where the PE is partitioned (vertically) into functional units of around 100–200 gates. The 5 wafer stack used only 2 different wafer types.

The 32 wafer stack contains 128x128 PEs using 5 different wafer types and a more flexible redundancy scheme. Rather than dedicating spares for each PE, spares are shared between 4 PEs, and PEs can be replaced by either of 2 spares. This flexibility provides good system yield with only 50% redundancy rather than the 100% of the earlier version. The actual yield on the 5 wafer prototype was 67%; this was predicted to fall to 3% using the same redundancy scheme on the larger design, or rise to over 90% with the more flexible scheme. Each PE supports floating point arithmetic in addition to the 16-bit fixed point of the earlier version. The die size is 2.5 in^2.

These systems run at around 10MHz, including the planned future version with 512x512 PEs on 4 inch dies.

Testing in the 32x32 system is straightforward: each functional unit is tested separately, and faulty ones are disconnected prior to the addition of the microspring bridge connection between them. The 128x128 system will use a YAG laser, integrated with the probe station. This larger system has a more complex redundancy scheme that requires switching between the spares. The primary is directly attached to one footpad of the microbridge, and either of the two spares can be switched in to operate at the other footpad.

Runtime reconfiguration was deemed sufficiently important to warrant the addition of special wafers to reroute vertically around faulty cells[10]. This vertical reroute scheme was preferred to expanding the functional units with soft-configurable logic partly so that it can operate without requiring a redesign of the 128x128 3-D computer.

6.3 Voting - Trilogy Triple Modular Redundancy

The highly publicized failure of Trilogy in 1984 has not encouraged further exploration in the direction of TMR (triple modular redundancy) as an approach to wafer scale integration[33].

Trilogy designed an IBM compatible mainframe aimed at the high end of the market, where extra cost can be justified by superior performance. They solved several major problems, designing CAD tools to assist in TMR placement and routing, and a special packaging scheme capable of removing $50W/cm^2$. This permitted the use of very high speed emitter-coupled logic (ECL) circuits that could dissipate over 1000W from the wafer.

With TMR, every logic unit is replicated 3 times, with circuitry to vote on the outputs, forwarding the majority result to other units. Trilogy's yield concerns led to the spacing of the three redundant copies at least 100 mils apart, greatly increasing the interconnect lengths. Without this spacing, a single cluster fault could destroy the whole wafer. With spacing, the long interconnects slowed down the signals.

Although functional wafers were produced, they were not cost effective. Redesign for higher wafer yield with acceptable performance would have taken too much time. TMR redundancy has been used successfully at higher levels (for example, the Tandem S2 computer uses a majority vote of 3 MIPS R2000 CPUs), but has yet to be shown worthwhile for high performance at the logic block level.

6.4 Hybrid systems

System density can be increased by packaging multiple dies together on a single substrate. Such a substrate can be regarded as a wafer (and could even be constructed from silicon) without the inconveniences of faulty components. While the dies cannot be as closely spaced as on a wafer, there is no need to bypass faulty sites, since all dies are pretested. Such a unit is referred to as a multichip module.

A myriad of approaches have been investigated, both by industry and by university researchers; these are overviewed in Swartzlander's book[47]. Two are briefly described here: the IBM modules which were the first to be used in a commercial product, over 10 years ago; and the DEC modules introduced in a recently announced product.

6.4.1 IBM's Thermal Conduction Module

IBM first introduced a multichip module in 1979 with the announcement of the 4300 series of processors[12]. This was enhanced to include more chips and wiring layers and to provide cooling in the thermal conduction module (TCM) technology introduced for the 3081 in 1982[7, 43, 2, 35]. The modules in the 4300 series contain 9 chips totalling around 4000 circuits with a total cooling capability of 9W. The TCM improved these figures to 118 chips with 25K logic circuits and 65K array bits and a total cooling capacity of 300W.

Each chip (0.457cm on a side with up to 704 circuits) is flip mounted onto the ceramic substrate, using solder bumps to connect anywhere within the die. Cooling is provided through spring loaded pistons that directly contact the back of the dies. The assembly is filled with helium to minimize thermal resistance at the chip to piston and piston to module interfaces. Total thermal resistance from chip surface to module exterior is at most 10°C/W. Each TCM is 15x15x6 cm^3. A printed circuit board was developed to support up to 9 TCMs, each connected to the board through up to 1800 pins.

Compared to the earlier design for the 4300 series, the TCM reduced total system wiring by an order of magnitude (2.4km to 305m) and off chip signal delay by a factor

of 4. However TCMs remain a costly alternative used only for high-end systems.

6.4.2 DEC's High-Density Signal Carrier

The HDSC technology was introduced with the VAX 9000 mainframes in 1989[16]. Chips are connected using TAB (tape-automated bonding) to a substrate containing 9 signal or power layers. These contain copper wiring and polyimide insulator (with a dielectric constant of 3.5) mounted on a copper base plate. The substrate is 4 inches on a side, and can hold up to 72 RAM chips, or 8 macrocell array chips (using the Mosaic III ECL process from Motorola). The substrate is mounted in 6 in^2 aluminum housing containing an air cooled heat sink below the copper base plate. This entire unit is inverted and placed on a printed circuit board which can hold up to 16 units. Connection to the PCB is through bumps located in the 1 inch rim around the 4 inch substrate. Signals and power from the substrate use flex circuits to connect to the rim.

Each chip is limited to 30W and 360 pins, and the module is limited to 300W. The copper lines have a pitch of 3 mils. A projected performance improvement of 2:1 over regular packaging was verified by measuring prototypes. This measure includes the delay through the chips (identical sets of 300 chips were used).

Chapter 7

Conclusion

Soft configuration as a viable approach to wafer scale integration has been demonstrated through the implementation of a pipelined memory system. There are several directions in which this work can be continued. One is to extend the yield models of chapter 5 to predict an optimal site size. We discuss other research possibilities for soft configurable techniques, which applications might be suited to them, and speculate on prospects for the future success of WSI. We conclude by reviewing the contributions of this thesis.

7.1 Choosing the size of a site

Figure 7.1 further extrapolates the site yields presented in chapter 4. Yields drop exponentially as expected when the site size increases. Yields for the large site sizes are optimistic since power line width, driver size, and decoder complexity are not scaled. These results can be used to estimate an optimal site size from Figure 7.2 under these assumptions:

- The optimal site size is the one that provides the largest amount of usable RAM on the wafer, taking into account the expected number of yielding sites.

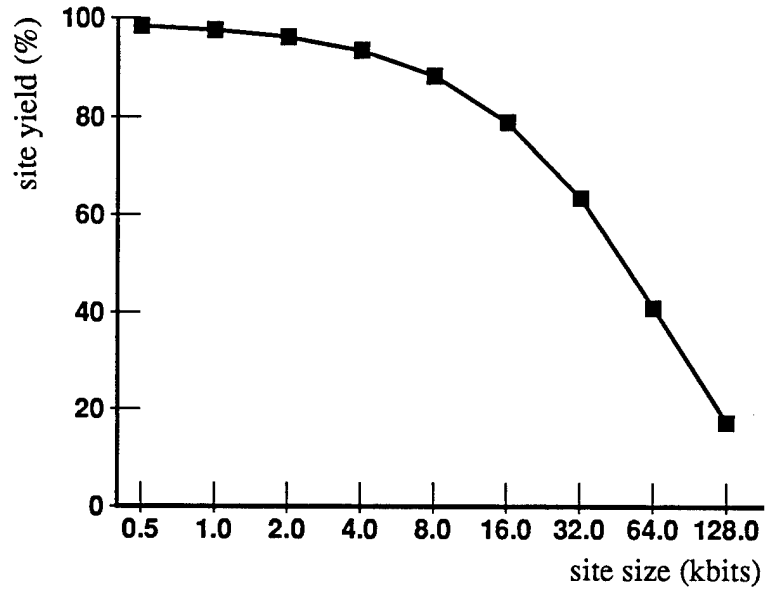


Figure 7.1: Site yield as a function of site size

- The amount of usable RAM is the number of working sites multiplied by the number of bits in each site. This is the baseline model described in chapter 5, with no assumptions about the distribution of working sites.

- Wafer area is held constant, at 10 cm by 10 cm.

- Switch area is held constant, with 3 different curves showing the effect of varying the switch size. The bottom curve assumes a switch of double width and height, the middle curve is the implemented switch size, and the top curve uses a switch that is half the width and height.

- Switch yield is high enough so that every working site can be accessed.

- Site area is scaled according to the size of the site components used for yield prediction. Site area is lower than is realistic for the large sites, since it omits the same effects as are omitted in the yield calculation.

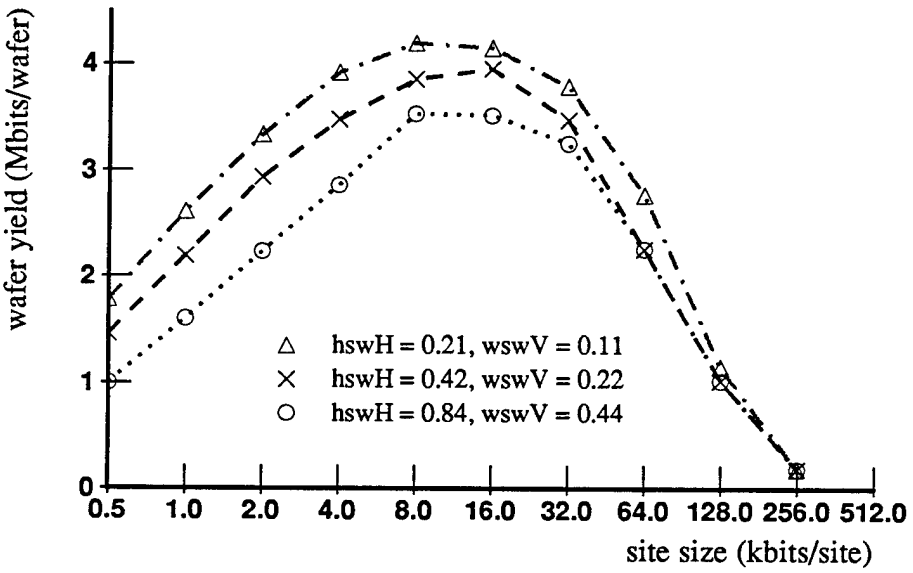


Figure 7.2: Maximizing available wafer memory as a function of site size. Switch size is held constant. hswH is the height between vertical neighbor sites. wswV is the width between horizontal neighbor sites. Switch dimensions are in millimeters. The implemented switch has hswH = 0.42 mm and wswV = 0.22 mm as shown in the middle of the 3 curves.

- The number of sites varies depending on how many of each size fit, with the appropriate number of switch widths in between.

Keeping the switch area constant is a fair approximation since word and address width change slowly relative to site size. It could be held constant by time-multiplexing longer data items. Increasing the width of the switch buses as necessary would not greatly increase the size of the switch, which is currently limited by the width of the power lines, not the data bus. However, the variations in switch size shown in Figure 7.2 change only the maximum amount of memory that can be gleaned from the wafer, not the optimal site size at which that maximum is obtained.

Figure 7.2 shows wafer yield increasing to a maximum for sites with 8 to 16 Kbits, before dropping rapidly for the larger sizes due to low site yields. Low wafer yield for very small sites results from the large proportion of the wafer that is devoted to switches. The Anamartic 32 Kbit dRAM unit is just to the left of the 8 Kbit SRAM with 48K transistors, assuming the dRAM design translates to roughly similar yields in an SRAM with the same number of transistors.

7.2 Prospects for soft configurable WSI

The pipelined memory implementation is a beginning in the development of soft configuration. Several issues remain to be resolved to continue this progress, and put it to work in a larger scale design. Some are specific to soft configurable WSI, others are considerations in any wafer scale design.

The switch network has input and output ports that must be connected to the rest of the system. The interface design depends on the constraints of the particular application. It will include circuitry to distribute inputs to the appropriate switch paths, to collect outputs, and to fit all of these inputs and outputs into a package with an appropriate level of cooling. Packaging and cooling are concerns in any WSI design, however the feeding and collection circuitry must be matched to the switch paths and timing of the soft configurable system. Power and clock distribution must also be addressed in any

design, but are particularly challenging for WSI.

Performance of the soft configurable switch network is an important issue that can be studied in part by simulation. While the actual switch delay depends on the circuit and technology, simulation can determine how many switch delays are incurred on average. The pipelined memory can be simulated using program address traces to measure the frequency of conflicts, and how much effect they have. They can be handled by dropping the data and relying on software resends, or by delaying the input a cycle or two, or by implementing congestion circuitry to hold the data inside the switch network until it can be safely delivered to the next stage.

If the soft configurable memory is used as a main memory responding to requests to fill cache blocks, then conflicts resulting from accesses to the same site can be eliminated by interleaving the memory, storing successive addresses in different sites. Software simulation can predict conflict frequencies, providing average latency and bandwidth to assist in choosing the number of switch paths and size of RAM sites best for the application. These choices may need adjusting to maximize wafer yield, if yield goals conflict with improving performance.

Yield estimates to guide a future design can be obtained by scaling down the 2μm mask layouts to current technology limits. The VLASIC yield simulation suite provides a program ("scalevl") to assist in this process. Experimenting with the defect statistics could show which are the most critical inputs to VLASIC. One such input is the defect distribution; modifying it to include wafer edge effects may be particularly important for designs that rely on a minimum number of functioning edge sites.

Predicting yield of the circuit components is important for any WSI design, and does not rely on anything specific to soft configuration. However, combining these yields in a meaningful way to predict global wafer yield is a process that does depend on the specifics of the redundancy method. The yield models presented in chapter 5 provide a starting point, and illustrate several important issues (such as the importance of very high switch yield, and the distribution of good sites). Their accuracy can be increased by including the likelihood of global signal failures, together with the probability that whatever mechanisms are available succeed in bypassing such failures when they occur.

The next version of VLASIC (currently in alpha release) includes a flag to count defects on a specific net, which can be listed in extensive detail giving the cause of each likely failure. This can be used to predict the probability of failure for each global signal.

Switch path configuration with bypasses profit by optimizations that eliminate unnecessary delays, coalescing multiple bypasses wherever possible. Such an optimizer used with the Monte Carlo simulation of the balanced load model can predict yield variation as a function of the maximum bypass delay.

Testing is particularly important for a low cost implementation of soft configuration. Probing each site and switch is expensive, so is best avoided. Our approach to soft configuration separates the sites from the switch network, so they can be tested independently after switch paths are established. However, test sequences are limited by the width of the switch bus. Sites must be designed so that their components can be independently tested. RAMs are ideal; each word can be independently read and written. Other applications such as broadband network switching will require particular attention to testing. Switches can be tested by rerouting through alternate paths, until the source of a discrepancy is established. Systematic tests of every direction through each switch are needed, both for power-up, and as regular checks to catch and reconfigure around runtime switch faults.

The prospects for success with soft configurable WSI depend on integrating it with a suitable application. The pipelined memory system described in chapter 3 was implemented as a straightforward example to demonstrate the potential of soft configurable systems. It could be used as main memory, with each column-pair containing consecutively addressed sites. The pipelined accesses then provide streamlined retrieval of data to refill cache lines. This streamlined retrieval requires enough working sites so that an integral number of cache lines can be supplied by each column-pair, and the total number of such usable sites exceeds a specified minimum. The sites may be unevenly distributed across the wafer, with some column-pairs supplying more cache lines than others.

Limiting switch paths to within a column-pair and designing for high yield are both steps that reduce any extra delay incurred to bypass faulty switches. Main memory or disk cache implementations are feasible with this approach, whereas the Anamartic spiral design at a similar cycle time is only fast when compared against disks. However,

providing input and output ports for multiple column-pairs, either by further circuitry on the wafer or by external chips, adds to the cost relative to the one port Anamartic design.

Wafer scale memory designs compete with more conventional RAMs, and must provide a significant edge to succeed. They can easily provide this edge in RAM size, and may do very well with bandwidth, but cannot match the single chip latency. It is a generally known principle that volume designs will always beat special purpose ones, unless the special purpose unit can provide a large advantage. This is because far more can be spent on designs for large volume markets. For this reason, the pipelined memory may be too specialized to compete with conventional RAM, and could be better suited to providing a test vehicle than a marketplace win for soft configurable WSI.

Manolis Katevenis has proposed a design for broadband network switching that can be implemented using soft configurable WSI[25]. By providing separate buffers for each virtual circuit (VC), a communication can be independent of congestion in other VCs. Large commodity dRAMS can provide enough memory for plenty of buffers, so that the number of VCs is not unduly limited. An example from [25] shows how to provide peak bandwidth of 4 Mbits/s for up to 65,000 VC's using 2 Mbytes per node plus 256 Kbytes/km.

Fair scheduling divides the bandwidth evenly amongst all ready VCs, while limiting the delay for those that are uncongested. This scheduling can be readily implemented using a network of multiplexing elements that choose which of the ready VCs to service. Each site in a wafer scale implementation would contain a multiplexing element with a few kilobits of buffer memory and some very simple logic, likely to fit in 2x2 mm^2. The VC buffer store would be a much larger external memory. A soft configurable wafer could hold a 16x16 or 32x32 switching node with a bandwidth of 1Gbit/s on each I/O link using internal 32-bit buses running at a cycle time of 25-30ns.

Broadband network switching is an interesting application that might benefit greatly from the soft configurable approach. However, this design could be implemented faster using conventional technology, since all of the wafer scale issues would then be side-stepped. Speed of implementation remains a crucial issue so long as technology continues to improve rapidly. The first one to market in the new technology wins, unless there are

serious design flaws. Simplicity helps to reduce design flaws, and provides less to design.

For these reasons, WSI is not likely to succeed except in specialized niches. One such niche could be the design of systems that require runtime reliability, if the redundancy can be leveraged for both goals. Such a system could be produced using soft configurable WSI. Only time will tell when the technology race will slow to the point that longer design times become profitable.

WSI research may not produce a successful whole wafer product in the near future, however the fault tolerance techniques can be applied on a smaller scale to increase die sizes. The INOVA large static RAMs are an example of this. The Anamartic wafer scale product is too recent for any conclusions, but if it succeeds, it will be due to improvements in reliability rather than cost or performance.

Exponential increases in lithography costs are likely to make multichip carriers cost effective by comparison; it is not clear that there will be much positive effect on WSI. Products that require very high reliability are likely to be the driving force behind any future successful WSI design, as this is one area in which WSI excels and can provide at least an order of magnitude improvement.

7.3 Review of contributions

A soft configurable pipelined memory system was designed together with Manolis Katevenis. The primary contributions of my work are the implementation and yield analysis of this design.

Soft configurable techniques have been demonstrated through the construction of working prototypes. These techniques include the use of switch buses both for site data and for switch control commands. Switch control is initiated by a redundant sequence that detects both stuck lines and bridging faults. Pipelining and multiple paths utilize the high bandwidth available within the wafer.

Yield analysis based on Monte Carlo fault simulations predict the yield of each circuit unit. Such simulations provide the best available prediction, as they are based directly

on layout masks. We combine component yields using several models to predict system yield for varying system requirements. These models show how system yield varies depending on the distribution of fault-free sites and switches. Finally, we summarize by predicting the optimal site size to maximize wafer yield in future designs.

Appendix A

Command Coding

Figure A.1 shows the switch and site command formats. Format (e) is the initial configuration sequence for setting up switch paths. Format (d) is the simple but unsafe version for enabling site outputs after the switch paths are configured and tested. Format (c) loads a site address. Format (b) initiates reads and writes using data specified according to format (a). Note that the rotated versions of the configuration sequence begin with a 0, so look like data to other switches and sites.

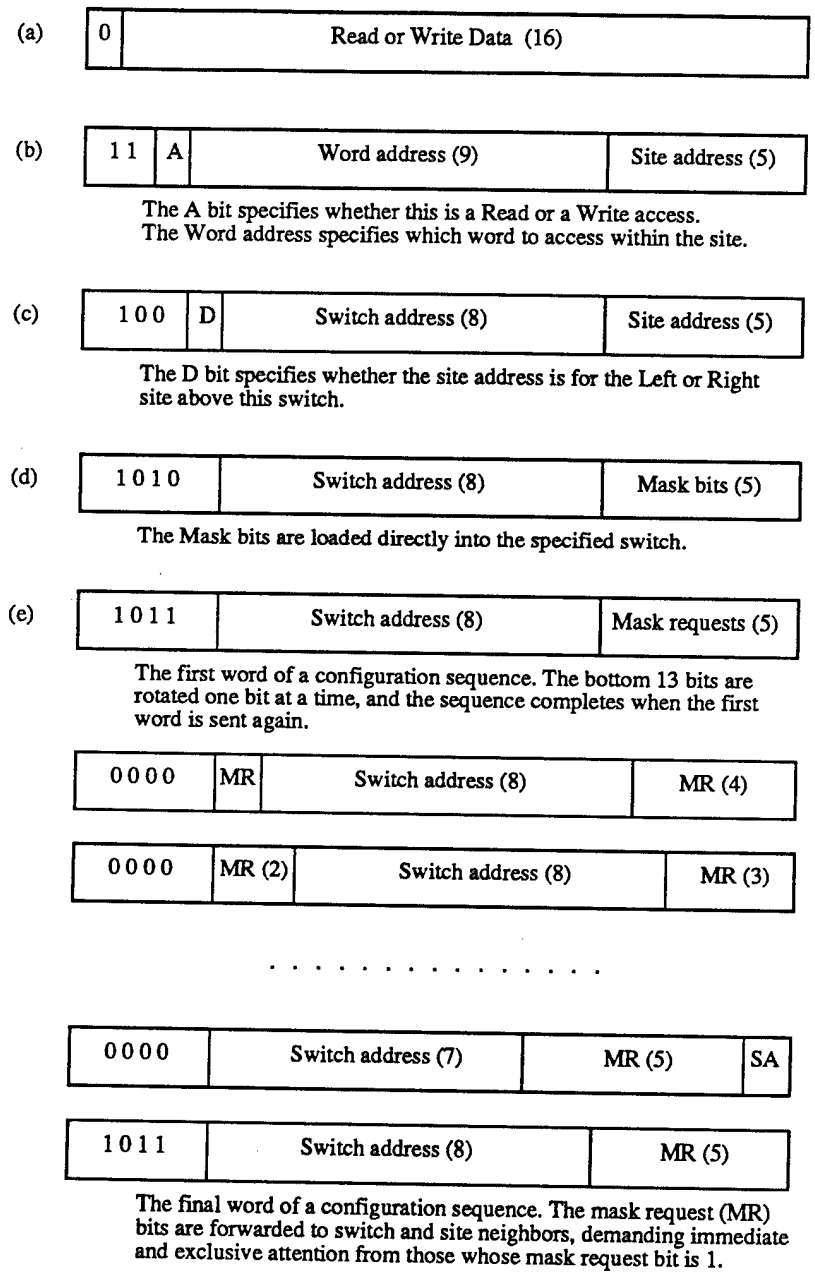


Figure A.1: Command formats. Field widths are shown in parentheses, or are 1 bit where omitted. (a)-(c) are site commands and data, (d)-(e) are switch commands.

Appendix B

16x16 switch addresses from 4 masks

The goal is to produce 256 different patterns at the corners between the 16x16 switches. With 4 different masks overlapping at each corner, these 256 patterns make up all 4^4 possibilities. We code each of the 4 masks with 2 bits, and refer to the placement of these masks to produce all 256 different overlapping patterns as the 2 bit solution.

The requirement that the 4 masks overlapping at each corner are unique corresponds to requiring that at least one of the components of each 2x2 subarray differs from the every other subarray. Thus, the equality

```
for x:= 0,1
    for y:= 0,1
        a[I + x, K + y] = a[II + x, KK + y]
```

holds for all four (x,y) pairs if and only if I = II and K = KK, where "a" is a 16x16 array of 2 bit entries, and I,II,K,KK all run from 0 to 14.

Here is the 1 bit solution which is a 4x4 array such that every 2x2 subarray is unique. Note that this holds even for patterns that wrap around edges and corners.

```
0 0 0 1
0 0 1 0
1 0 1 1
0 1 1 1
```

We produce the 2 bit solution in a 16x16 array by copying the 1 bit solution to every 4x4 subarray. The first of the 2 bits is exactly the 1 bit solution as shown above, and the second bit is from the same solution, but rotated (using wraparound) to start from a different position.

Figure B.1 shows a 2 bit solution with every 2x2 subarray unique, found by trial and error[17]. It is given first by the starting locations of the 1 bit solution for each 4x4 subarray used in the second bit. The indices here start from (0,0) at the top left, with the first one giving the row, and the second giving the column. For example, the subarray at the top right has starting location (0,0), so in the full solution below, that subarray is not rotated, and every pair of bits is identical. It was checked using a simple program to compare all the 2x2 subarrays with one another.

(0,1)	(1,1)	(3,0)	(0,0)
(0,2)	(1,0)	(1,3)	(2,1)
(3,2)	(2,0)	(0,3)	(3,1)
(3,3)	(2,3)	(2,2)	(1,2)

I:	K: 0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15
0	00 00 01 10	00 01 00 10	00 01 01 11	00 00 00 11
1	00 01 10 00	00 01 11 01	00 00 10 01	00 00 11 00
2	10 01 11 11	11 01 11 10	10 00 11 10	11 00 11 11
3	01 11 11 10	00 10 11 10	01 10 11 11	00 11 11 11
4	00 01 00 10	00 00 01 10	00 00 00 11	00 01 01 11
5	01 00 10 00	01 00 11 01	01 01 10 01	01 01 11 00
6	11 01 11 10	10 01 11 11	11 00 11 11	10 00 11 10
7	01 11 10 11	00 10 10 11	01 10 10 10	00 11 10 10
8	01 01 00 11	01 00 01 11	01 00 00 10	01 01 01 10
9	00 01 10 00	00 01 11 01	00 00 10 01	00 00 11 00
10	11 00 10 10	10 00 10 11	11 01 10 11	10 01 10 10
11	01 11 11 10	00 10 11 10	01 10 11 11	00 11 11 11
12	01 00 01 11	01 01 00 11	01 01 01 10	01 00 00 10
13	01 00 10 00	01 00 11 01	01 01 10 01	01 01 11 00
14	10 00 10 11	11 00 10 10	10 01 10 10	11 01 10 11
15	01 11 10 11	00 10 10 11	01 10 10 10	00 11 10 10

Figure B.1: 2 bit solution for 16x16 array of 4 different masks. The top version shows the starting location of the 1 bit solution for the second bit in each 4x4 subarray. The bottom version shows the entire 16x16 array of 2 bits.

Bibliography

[1] John Michael Acken.
Deriving accurate fault models.
PhD thesis, Stanford University, Department of Electrical Engineering, 1988.

[2] R.C. Chu amd U.P. Hwang and R.E. Simons.
Conduction cooling for an LSI package: A one-dimensional approach.
IBM Journal of Research and Development, 26(1):45–54, January 1982.

[3] A.H. Anderson, R. Berger, K.H. Konkle, and F.M. Rhodes.
RVLSI applications and physical design.
In *IEEE International Conference on Wafer Scale Integration*, pages 39–46, January 1989.

[4] R. Aubusson and I. Catt.
Wafer-scale integration — a fault-tolerant procedure.
IEEE Journal of Solid-State Circuits (JSSC), SC-13(3):339–344, June 1978.

[5] Fumio Baba and Alan Sinclair.
200-Mb wafer scale memory.
In *IEEE International Conference on Wafer Scale Integration*, pages 5–12, January 1990.

[6] R. Berger, A. Bertapelli, R. Frankel, J.J. Hunt, J. Mann, J.I. Raffel, F.M. Rhodes, A. Soares, and C. Woodward.
The Lincoln programmable image-processing wafer.
In *IEEE International Conference on Wafer Scale Integration*, pages 20–26, January 1990.

[7] A.J. Blodgett and D.R. Barbour.
Thermal conduction module: A high-performance multilayer ceramic package.
IBM Journal of Research and Development, 26(1):30–36, January 1982.

[8] R. Bourassa, T. Coffman, and J.E. Brewster.
Progress in WSI SRAM development.
In *IEEE International Conference on Wafer Scale Integration*, pages 13–19, January 1990.

[9] Donald F. Calhoun and George Wolfe.
Status and future directions of pad relocation full wafer LSI.
In *Proceedings 1973 Electronic Components Conference*, pages 7–14, 1973.

[10] M. Campbell, M. Little, and M. Yung.
Hierarchical fault tolerance for 3D microelectronics.
In *IEEE International Conference on Wafer Scale Integration*, pages 174–188, January 1990.

[11] G.H. Chapman, J.M. Canter, and S.S. Cohen.
The technology of laser formed interactions for wafer scale integration.
In *IEEE International Conference on Wafer Scale Integration*, pages 21–30, January 1989.

[12] B.T. Clark and Y.M. Hill.
IBM multichip multilayer ceramic modules for LSI chips — designs for performance and density.
IEEE Transactions on Components, Hybrids and Manufacturing Technology, CHMT-3:89–93, 1980.

[13] Earl E. Swartzlander, Jr, editor.
Wafer Scale Integration.
Kluwer, 1989.

[14] R. Frankel, J.J. Hunt, M. Van Alstyne, and G. Young.
SLASH — an RVLSI CAD system.
In *IEEE International Conference on Wafer Scale Integration*, pages 31–38, January 1989.

[15] William A. Geideman and Allen L. Solomon.
Wafer integrated semiconductor mass memory.
In *International Telemetry Conference*, pages 955–962, November 1978.

[16] Susan Godsell Baust, Richard J. Dischler, and Scott Westbrook.
Implementing a packaging strategy for high-performance computers.
High Performance Systems, pages 28–31, January 1990.

[17] Barry Hayes.
16x16 puzzle with 4 corner patterns.
Private communication, December 1985.

[18] Kye S. Hedlund.
WASP — a wafer-scale systolic processor.
In *Proceedings of the IEEE International Conference on Computer Design*, pages 665–671, 1985.

[19] Kye S. Hedlund.
The design of a prototype WASP machine.
In G. Saucier and J. Trilhe, editors, *IFIP Workshop on Wafer Scale Integration*, pages 89–97, March 1986.

[20] Kye S. Hedlund and Lawrence Snyder.
Systolic architecture — a wafer scale approach.
In *Proceedings of the IEEE International Conference on Computer Desi gn*, pages 604–610, 1984.

[21] John L. Hennessy and David A. Patterson.
Computer Architecture: A Quantitative Approach, pages 579–582.
Morgan Kaufman, 1990.

[22] Y. Hsia, G. Chang, and F.D. Erwin.
Adaptive wafer scale integration.
In *Digest of Technical Papers, of the 11th Conference on Solid State Devices, Tokyo*, pages 81–82, 1979.

[23] Barry W. Johnson.

Design and Analysis of Fault Tolerant Digital Systems.
Addison-Wesley, 1989.

[24] J.M. Kallis, L.B. Duncan, S.P. Laub, M.J. Little, L.M. Miani, and D.C. Sandkulla.
Reliability of the 3-D computer under stress of mechanical vibration and thermal cycling.
In *IEEE International Conference on Wafer Scale Integration*, pages 65–72, January 1989.

[25] M. G. H. Katevenis.
Fast switching and fair control of congested flow in broadband networks.
IEEE Journal on Selected Areas in Communications, SAC-5(8):1315–1326, October 1987.

[26] M. G. H. Katevenis and M. G. Blatt.
Switch design for soft-configurable WSI systems.
In *1985 Chapel Hill Conference on Very Large Scale Integration*, pages 197–219, Chapel Hill, North Carolina, March 1985.

[27] J. W. Lathrop.
Discretionary wiring approach to large scale integration.
In *WESCON/66 Technical Papers*, volume 10/3, pages 1–6, 1966.
Part 2.

[28] Michael J. Little and Jan Grinberg.
The 3-D computer: An integrated stack of wsi wafers.
In Earl E. Swartzlander, editor, *Wafer Scale Integration*, chapter 6, pages 253–318. Kluwer, 1989.

[29] M.J. Little, R.D. Etchells, J. Grinberg, S.P. Laub, J.G. Nash, and M.W. Yung.
The 3-D computer.
In *IEEE International Conference on Wafer Scale Integration*, pages 55–64, January 1989.

[30] W. Lukaszek.
Yield statistics.
Private communication, 1988.

[31] Wojciech Maly.
Computer-aided design for VLSI circuit manufacturability.
Technical Report CMUCAD-89-39, SRC-CMU Research Center for Computer-Aided Design, 1989.

[32] F. Manning.
An approach to highly integrated, computer-maintained cellular arrays.
IEEE Transactions on Computers, C-26(6):536–552, June 1977.

[33] J. F. McDonald, E. Rogers, K. Rose, and A. Steckl.
The trials of wafer-scale integration.
IEEE Spectrum, 21(10):32–39, October 1984.

[34] Irwin Miller and John E. Freund.
Probability and Statistics for Engineers.
Prentice-Hall, 1965.

[35] S. Oktay and H.C. Kammerer.
A conduction-cooled module for high-performance LSI devices.
IBM Journal of Research and Development, 26(1):55–66, January 1982.

[36] Fabian Pease.
Cost of wafer exposure tool as a function of year.
Private communication, May 1990.

[37] Richard L. Petritz.
Current status of large scale integration technology.
In *IEEE Journal of Solid-State Circuits*, volume SC-2/4, pages 130–147, December 1967.

[38] Jr. R. M. Warner.
Applying a composite model to the IC yield problem.
IEEE Journal of Solid-State Circuits, SC-9(3):86–95, June 1974.

[39] Jack Raffel, Allan H. Anderson, and Glenn H. Chapman.
Laser restructurable technology and design.

In Earl E. Swartzlander, editor, *Wafer Scale Integration*, chapter 7, pages 319–364. Kluwer, 1989.

[40] G. Saucier, J-L Patry, A. Boubekeur, and E. Sanlaville.
Practical experiences in the design of a wafer scale 2D array.
In Vijay Jain, editor, *Defect and Fault Tolerance in VLSI Systems*, volume 2. Plenum Press, 1990.

[41] G. Saucier, J-L Patry, and E-F Kouka.
A reconfigurable wafer scale array for image processing.
In *IEEE International Conference on Wafer Scale Integration*, pages 277–288, January 1989.

[42] G. Saucier, J-L Patry, E-F Kouka, T. Midwinter, P. Ivey, M. Huch, and M. Glesner.
Defect tolerance in a wafer scale array for image processing.
In Israel Koren, editor, *Defect and Fault Tolerance in VLSI Systems*, volume 1, pages 327–338. Plenum Press, 1989.

[43] Donald P. Seraphim.
A new set of printed-circuit technologies for the IBM 3081 processor unit.
IBM Journal of Research and Development, 26(1):37–44, January 1982.

[44] Charles H. Stapper.
The effects of wafer to wafer defect density variations on integrated circuit defect and fault distributions.
IBM Journal of Research and Development, 29(1):87–97, January 1985.

[45] D. M. H. Walker.
Yield simulation for integrated circuits.
Technical Report CMU-CS-86-143, Carnegie Mellon, Department of Computer Science, 1986.

[46] B. Warren, W. Richardson, K. Kanegawa, and C. Arnell.
A one megabit SRAM fabricated with 1.2μm technology.
In *IEEE International Conference on Wafer Scale Integration*, pages 47–54, January 1989.

[47] R. Wayne Johnson, Richard C. Jaeger, and Travis N. Blalock.
Wafer-scale multichip packaging technology.
In Earl E. Swartzlander, editor, *Wafer Scale Integration*, chapter 10, pages 473–500. Kluwer, 1989.

[48] P.W. Wyatt and J.I. Raffel.
Restructurable VLSI — a demonstrated wafer scale technology.
In *IEEE International Conference on Wafer Scale Integration*, pages 13–20, January 1989.

[49] M.W. Yung, M.J. Little, R.D. Etchells, and J.G. Nash.
Redundancy for yield enhancement in the 3-D computer.
In *IEEE International Conference on Wafer Scale Integration*, pages 73–82, January 1989.